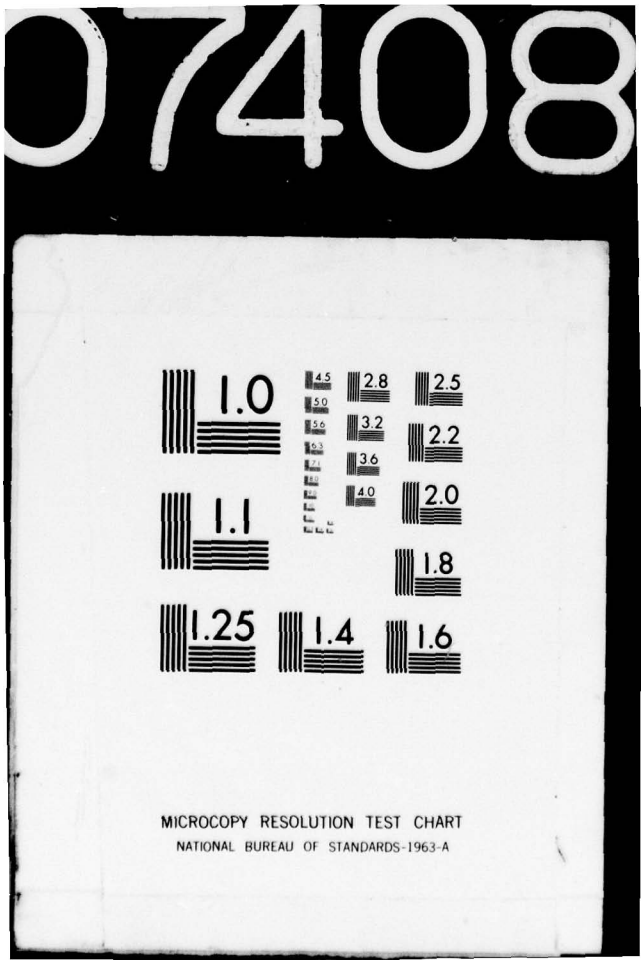
AD-A074 081 PLANNING RESEARCH CORP MCLEAN VA F/G 5/2
INVERTED FILE FACILITY USER'S GUIDE.(U)
AUG 77 S TAMBERRINO DCA100-73-C-0015
UNCLASSIFIED PRC-706-22 NL

1 OF 2
AD
A074081


07408
1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD A074081

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: PRC 706-22 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): (6) Inverted File Facility User's Guide		5. TYPE OF REPORT & PERIOD COVERED: User's Guide - Final
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): (10) S. Tamberrino		8. CONTRACT OR GRANT NUMBER(s): (15) DCA100-73-C-0015 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Planning Research Corporation, Data Services Company, 7600 Old Springhouse Rd, McLean, VA 22101 (408424)		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS: Command and Control Technical Center, Code C440, 11440 Isaac Newton Square (N), Reston, VA 22090		12. REPORT DATE: August 31, 1977 13. NUMBER OF PAGES: 93
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): (11) 31 Aug 77		15. SECURITY CLASS. (of this report): UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report): Unlimited (9) Final Rept.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report): Unlimited (12) 97 p.		
18. SUPPLEMENTARY NOTES: (14) PRC-706-22		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Inverted File Facility (INVFF) is an extension of the Indexed Sequential Processor (ISP) which allows the user to build, retrieve from, and update inverted data bases. In addition to the ISP's data file and index file, the INVFF uses an inverted index file which provides the following additional file processing capabilities: (1) the ability to distinguish each data record as being a specific type and (2) the ability to process data records randomly through the specification of values contained within a record.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE 402 560

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

LEVEL II

12
B.S.

INVERTED FILE FACILITY
USER'S GUIDE

See 1473 in back.

Contract No: DCA 100-73-C-0015
Technical Report: PRC 706-22
August 31, 1977
S. Tamberrino

PRC DATA SERVICES COMPANY

CONTENTS

CONTENTS

SECTION I

GENERAL DESCRIPTION

The Inverted File Facility (INVFF) is an extension of the Indexed Sequential Processor (ISP) which allows a user to build, retrieve from, and update inverted data bases. An inverted data base consists of three separate files, a data file, an ISP index file, and an inverted index file. The data file and the ISP index file are identical in format to the data file and index file in an indexed sequential data base. The inverted index file provides for the inverted data base some file processing capabilities which do not inherently exist for an indexed sequential data base. These capabilities are:

1. The ability to distinguish each data record as being of a specific type. The record type is determined by the inverted file processor by examining the content of a data field within the record. This data field is known as the record type field and is defined in the inverted index file. The definition of the record type field may be omitted, in which case all of the records on the file are defined as being of the same type.

2. The ability to process data records randomly through the specification of values contained in one or more data fields within a record. These data fields are known as inverted key fields and are defined in the inverted index file. One or more inverted key fields may be defined for each record type.

   An inverted key is the logical association of one or more inverted key fields; inverted key fields from different record types may be defined as participating in the same key, allowing those fields to be processed equivalently with regard to retrieval via specification of a given value. It is possible for a data field to participate in more than one key.

   Any number of keys may be defined for a particular record type, and any number of record types can contain any one particular key.

   The inverted index file contains the definition for each inverted key and one or more index pages for each key. The index pages for each key contain all unique values of that key which occur in the data file and the data file address of every record which contains each key value.

In addition to the inverted keys, an inverted data base has a primary or ISP key. The ISP key is defined and maintained on the ISP index file and serves the same function that is serves for a simple indexed sequential data base.

USER INTERFACE

The user interface to the Inverted File Facility is composed of a set of descriptor cards (which are placed by the user on a descriptor card file for access by the INVFF), a set of subroutines, and the utility program XUTIL.

Descriptor Cards

The descriptor cards contain the parameters that define the characteristics of the inverted file. The types and functions of the descriptor cards are listed below. Specific details concerning their formats and usage are contained in Sections II and III.

Card Type	Function
INDEX	Specifies GCOS file code and page size of the ISP index file.
DATA	Specifies characteristics of the data file.
INDX2	Specifies GCOS file code, page size, and percentage fill for the inverted index file.
RECORD	Specifies the characteristics of the largest record in an inverted data file.
WORKFL	Specifies the file code of the temporary file on which record addresses from the inverted index file are to be written during a retrieval job. The file must be a random file.
SORTFL	Specifies the file code of the temporary file on which entries for the inverted index file are to be written during a build job. The file must be a magnetic tape or linked disk file.
KEY	Specifies the characteristics of a data field to be used as an inverted key field.
ETC	Continuation of a descriptor card of any other type.

Subroutines

The inverted file processing subroutines are invoked via the standard CALL interface. Refer to the individual programming reference manuals for specific CALL interfaces. The names and functions of the subroutines are listed below. Specific details concerning their usage are contained in Section II and IV.

Subroutine	Function
INVBLD	Allows inverted file build routines to be loaded.
KINIT	Initializes an inverted file. Designed primarily for use by World Wide Data Management System (WWDMS).
KOPEN	Opens a pre-initialized inverted file to prepare for the storage of records into the file for the first time. Designed primarily for use by WWDMS.

Subroutine	Function
NOPEN	Opens a new inverted file to prepare for the storage of records into the file for the first time.
FILINT	Stores a record into an inverted file that has been opened for building by NOPEN.
IOPEN	Opens an existing inverted file to prepare for the retrieval, addition, deletion, and modification of records in the file.
LOPEN	Accomplishes the same function as IOPEN. Designed primarily for use by WWDMS.
GETSEQ	Retrieves the next available record in sequence from the data file, without reference to the ISP or inverted index file.
GETRAN	Retrieves the record having the ISP key value equal to that specified by the user.
GETIND	Retrieves the data file addresses of all records containing inverted key values which meet criteria specified by the user.
GETREC	Retrieves a record having a data file address retrieved by GETIND.
XREWND	Rewinds the data file to either the first record in the file or the record having an ISP key value that is greater than or equal to a specified value.
INVUPD	Allows inverted file update routines to be loaded.
ADDREC	Adds a new record to an existing inverted file.
CHGREC	Replaces an existing record in an inverted file with a record having the same size and ISP key value.
DELREC	Deletes, from an existing inverted file, the record having the ISP key value specified by the user.
XSTAT	Stores the current values of seven different ISP file utilization parameters in an area specified by the user.
XFLUSH	Writes all pages that have been modified since they were last read from random access storage back to random access storage.
ICLOSE	Terminates processing of an inverted file that was opened by NOPEN, IOPEN, KOPEN, or LOPEN.

Subroutine	Function
WRAPUP	Terminates processing of all inverted and indexed sequential files that are currently open.
XCKPTF	Checkpoints all open files when FMS ABORT/ROLLBACK/ is specified.
XCKPTP	Checkpoints all open files and the program when FMS ABORT/ROLLBACK/ is specified.
XROLBF	Cancels all changes made to files since the last checkpoint was taken when the files are protected using the FMS ABORT/ROLLBACK/ option.
XROLBP	Restores file contents and the program to their state at last checkpoint when files are protected with the FMS ABORT/ROLLBACK/ option.

XUTIL Utility

The XUTIL utility is executed as a GCOS PROGRAM activity, with all necessary information being supplied by the user on control cards and descriptor cards. XUTIL provides the following inverted file processing capabilities:

1. Loading a new inverted file from an existing sequential file.

2. Unloading an inverted file to remove any deleted records.

3. Reloading an inverted file to redistribute the records or to add, delete, or modify definitions of inverted keys.

XUTIL is described in detail in Section II under the subsection on "UNLOADING AND RELOADING AN INVERTED FILE".

SECTION II

FILE PROCESSING

This section contains, in general, the information needed to use the Inverted File Facility in a batch processing environment. There are three basic modes of inverted file processing; building a new file, retrieving records from an existing file, and updating an existing file. Retrieval and updating can be done in the same program without difficulty, but each mode is discussed separately in this section.

BUILDING A NEW FILE

An inverted file is built initially by one call to INVBLD, one call to NOPEN, one call to FILINT for each record to be stored in the file, and one call to ICLOSE. The building of a new file may be broken down into two phases; the file initialization phase and the file loading phase.

The file initialization phase begins when the call to NOPEN is executed, and ends when control is returned by NOPEN to the user program. During this phase, the following processing takes place on the inverted indexed file.

1. The first control page is initialized. The control pages are those pages of the inverted index file which contain file management and utilization statistics and provide a means for accessing the key definition pages and the coarse and file index pages for each key.

2. The inverted file processor reads the key definitions from the descriptor card file, builds a key definition table in core, and then writes the entries from that table to one or more key definition pages on the file. The inverted file processor also writes an entry to the control pages for each key defined.

Some initialization of the data file and the ISP index file also takes place during the initialization phase.

The file loading phase begins when the first call to FILINT is executed and ends when control is returned from ICLOSE to the user program. Each time FILINT is called, the data record in core is stored on the data file. The inverted processor then determines the record type of the record. For each inverted key defined for that record type, the inverted file processor writes to a temporary sort file a sort entry consisting of the key number, the value contained in the key field in the data record, and the data file address of the record.

When the file is closed by ICLOSE, the sort entries are sorted on key number and key value. The inverted file processor then, for each key defined for the file, obtains from the inverted index file a fine index page and writes to that page each key value present on the file and the data file address of each record which contains that value.

If more than one fine index page is required to contain all of the key values and data file addresses relevant to a given key, the key values and data file addresses are overflowed onto one or more subsequent fine index pages, and an entry to each fine index page is written to a coarse index page. When a coarse index page overflows, entries for that coarse index page and the pages onto which it has overflowed are written onto a higher level coarse index page.

Descriptor Cards Required

The chart below shows the types of descriptor cards which must be present on the descriptor card file for a build program, the meaningful parameters which may appear on these descriptor cards, and an indication as to whether these parameters are optional or required.

Card Type	Parameters
INDEX	FC = isp-index-file-code PAGESZ = isp-index-page-size (optional, default = 320)
DATA	FC = data-file-code-n, n = 1,2,...,8 PAGESZ = data-page-size (optional, default = 320) PAGEFIL = pct-fill (optional, default = 100) COLCOM (optional)
RECORD	RECSZ = record-size KEYSZ = key-size KEYOFF = key-offset (optional, default = 0) RTYPSZ = record-type-size (optional) RTYPOFF = record-type-offset (optional, default = 0)
INDX2	FC = invff-index-file-code PAGESZ = invff-index-page-size (optional, default = 320) PAGEFIL = invff-pct-fill (optional, default = 90)
SORTFL	FC = sort-file-code
KEY	RTYPE = record-type (required if an only if RTYPSZ is present on RECORD card) KEYNO = key-number KEYSZ = key-size KEYOFF = key-offset (optional, default = 0) NULL = { null-character / BLANK } (optional) UNIQUE (optional) SIGNED (optional)

NOTES

1. There must be one KEY card for each inverted key for each record type for which that key is to be defined.

2. If no record type field is to be defined for the file, the RTYPSZ and RTYPOFF parameters should be omitted from the RECORD card, and the

RTYPE parameter should be omitted from each KEY card. This will cause all inverted keys defined for each record on the file.

3. If a record type field is defined, there must be at least one inverted key defined for each type of record that is to be stored on the file.

For a more detailed description of descriptor cards and descriptor card parameters, refer to Section III.

File Build Subroutines

Below is a summary of the subroutine calls needed in an inverted file build procedure.

1. One call or SYMREF to INVBLD - The subroutine itself accomplishes nothing, but the call or SYMREF must be present anywhere in the user program in order to cause the General Loader to load the inverted file build modules. No calling arguments are necessary.

2. One call to NOPEN to initialize the file and prepare for the storage of records into the file for the first time by FILINT. The syntax of the call to NOPEN is as follows:

```
CALL NOPEN USING record-area, data-file-code-1,
     desc-file-code [,error-code]
```

3. One call to FILINT for each record to be stored in the file. The syntax of the call to FILINT is as follows:

```
CALL FILINT USING data-file-code-1 [,alt-record-size
     [,error-code, seq-check]]
```

4. One call to ICLOSE to terminate processing of the file. The syntax of the call to ICLOSE is as follows:

```
CALL ICLOSE USING data-file-code-1
```

NOTES ON CALLS

1. ISP assumes that the records being stored by FILINT are in the proper key value order according to the collating sequence selected by the user. Optionally, the user may request that the key value sequence be checked for errors.

2. If the user attempts to store via FILINT a record containing an inverted key value that is a duplicate of a value already present on the file, and if that inverted key was defined as UNIQUE, then the record will be stored in the data file but no entry for the duplicate occurrence will be stored in the inverted index file. A warning message will be printed on the utilization report.

3. Each record to be stored by FILINT must be large enough to contain all of the inverted keys defined for the record in their data record orientation.

For a more detailed description of the usage of these subroutines, refer to Section IV.

Temporary Files Needed

When building a new inverted file, the user is required to provide his program with two temporary files: a sort file and a sort collation file.

SORT FILE

The sort file is the file on which the inverted processor stores entries containing key values and pointers while the data file is being loaded via FILINT. The sort file must be either a magnetic tape or linked disk file. If a linked disk file is to be used, it is advisable to allocate as many links for the sort file as is occupied by the inverted index file.

SORT COLLATION FILE

When ICLOSE is called in a build routine, the Sort/Merge utility program is called in to sort the entries which have been placed on the sort file. The Sort/Merge requires that one or more collation working files be allocated by the user.

The collation file(s) may be assigned to either magnetic tape or random disk. If the sort file is a linked disk file, it is advised that the sort collation file be a random disk file of the same size, assigned a file code of S1. If the sort file is a magnetic tape file, then three tapes, assigned file codes of S1, S2, and S3, should be allocated as sort collation files.

Restricted File Codes

In a program, such as an inverted file build program, in which the Sort/Merge Program is executed, all file codes beginning with the letter S are restricted to files directly involved in the Sort or Merge process. Thus, in an inverted file build program the use of any file code beginning with the letter S should be avoided, except with regard to the sort collation files.

Memory Assignment Manipulation

To reserve memory space for the Sort/Merge program to use as a work area, one of the following sets of control cards must be included in the job setup:

```
1.  $    LOWLOAD    (optional)
    $    USE        .SMA/1/,.SMB/ssize/,.SMC/1/

    or

2.  $    LOWLOAD
    $    USE        .XBUF1/xsize/,.XBUF2/2/
```

or

3. $ USE .XBUF/2/,.XBUF1/xsize/

where:

ssize = 8000 memory words or more
xsize = 4000 memory words or more

If option (1) is used, the greater the value of ssize the more efficiently the Sort/Merge program will operate. However, a larger value for ssize also means that the inverted file processor will have less memory available for use as buffer space; therefore, the amount of I/O performed by the inverted processor will increase. If option (2) is used, the reverse is true with respect to the value of xsize.

File Size Computations

The formulas to calculate the size of the data file and ISP index file are described in the current ISP manual.

The required size of the inverted index file is difficult to calculate; this quantity is dependent upon such variables as the number of record types defined for the file, the number of inverted keys defined for each record type, the size of the inverted key fields, and the composition of the collection of records on the data file with respect to record type. However, the formulas given below can be used to produce a fairly reliable upper bound. The independent variables used in these formulas are:

NKU = Total number of key values in the file for unique keys

NKN = Total number of key values in the file for non-unique keys

IIPS = Inverted file index page size (in words)

PF = Percent fill

KSU = Key size in words (size in characters divided by 6 and rounded up to nearest whole number) of largest unique key

KSN = Key size in words of largest non-unique key

Additionally, let function INT n = largest integer less than or equal to n.

1. The number of occurrences of a unique key that will fit on an index page, KPU, is given by:

$$KPU = INT \left[\frac{INT \left[\frac{IIPS*PF}{100} \right] - 4}{KSU+1} \right]$$

2. The number of occurrences of a non-unique key that will fit on an index page, KPN, is given by:

$$KPN = INT\left[\frac{INT\left[\frac{IIPS*PF}{100}\right]-4}{KSN+2}\right]$$

3. The number of inverted pages required to hold the unique inverted keys, NDPU, is given by:

$$NDPU = INT\left[\frac{NKU}{KPU}\right]$$

4. The number of inverted pages required to hold the non-unique inverted keys, NDPN, is given by:

$$NDPN = INT\left[\frac{NKN}{KPN}\right]$$

5. The total number of inverted index pages required, NDP, is given by:

$$NDP = NDPU + NDPN$$

6. The number of 320-word blocks required for the inverted index file is given by:

$$NIB = INT\left[\frac{\left[\frac{IIPS*NDP}{64}\right]+4}{5}\right]$$

7. The number of 3840-word links required for the inverted index file, NIL, is given by:

$$NIL = INT\left[\frac{NIB+11}{12}\right]$$

Efficiency Techniques

1. Maximum page buffer utilization occurs when the inverted index page size, ISP index page size, and data page size are equal.

2. For a given inverted index file, increasing the page size of the file will decrease the overall size, as fewer coarse index pages will be required.

3. If the primary retrieval path to the data base is via a unique key, it is better to use smaller inverted index pages, since inverted index pages must be searched sequentially for a particular key value and more processing is required to search through a large page than a small page. If, however, the primary retrieval path is via a non-unique key for which many duplicate values are anticipated, the data file addresses corresponding to a given key value may well extend across several pages. In a case such as this, it is better to use larger page sizes in order to decrease the amount of inter-page processing.

4. If it is anticipated that a given key field may, in a majority of the records, contain no useful information, the use of the null processing option will result in a saving of both file space and processing time.

5. When inverted index entries and data file addresses are added to the inverted index file during an update procedure, they must be physically inserted. If there is no room on the page on which the entry or data file address is to be inserted, then that page must be split into two pages. Therefore, it is advantageous to reserve space for future expansion. The inverted file processor will, by default, reserve ten percent of the words on each page for this purpose. If the data base is to be used in a transaction processing environment, it may be beneficial to reserve more, via the PAGEFIL option on the INDX2 card.

Other efficiency techniques may be found in the current ISP manual.

RETRIEVING RECORDS FROM AN EXISTING FILE

There are three methods of retrieving records from an inverted file:

1. Sequentially

2. By specification of an ISP key value

3. By specification of a value, a set of values, or a range of values for each of one or more inverted keys.

Sequential retrieval and retrieval by specification of ISP key value do not involve the use of the inverted index file and are performed in the same manner with respect to inverted and indexed sequential files alike. These methods of retrieval and the procedures for using them are discussed in detail in the current ISP manual.

Records are retrieved by specification of inverted key values by one or more calls to GETIND, to retrieve the data file addresses of the records satisfying the user-specified criteria; and one call to GETREC for each data file address retrieved, to retrieve the record having that address in the data file.

The work file is a temporary file on which data file addresses are placed by GETIND and from which they are retrieved from GETREC. Prior to each call to GETIND, the work file may or may not contain data file addresses retrieved by one or more previous calls to GETIND. The user passes an option code to GETIND to specify what is to be done with this existing list of addresses when the new list is retrieved. If option code is set to one, the existing list of addresses is discarded. If option code is set to two or three, the existing list of addresses is saved for logical manipulation with the new list.

GETIND obtains the criteria for building a new list of record addresses by reading a list of conditionals passed by the user. A conditional is a true-or-false statement about the contents of an inverted key field of a

record. Each conditional is represented in the program logic as a three-element vector consisting of a key number, a relational code (a number corresponding to "greater than", "greater than or equal to", "equal to", "less than or equal to", or "less than"), and a specific key value to which the key value in each record is to be compared. Each conditional is used as input to an index searching process, and the output from this process is a list of data file addresses of records satisfying that conditional. When all conditionals have been processed, the resultant lists are consolidated into one list consisting of the data file addresses of the records which satisfy all of the specified criteria.

If a value of two or three was specified for the option code, the list of record addresses retrieved by the current call to GETIND, known as the active list, is manipulated with the list of addresses which was present on the work file prior to the current call, leaving a single list on the work file as control is returned to the user program. If option code has a value of two, then the resultant list will be a logical "or" of the active list and the previous list; that is, a list of all addresses which are present in either or both lists. If option code has a value of three, the resultant list will be a logical "and" of the active list and the previous list; that is, a list of all addresses which are present in both lists. Thus, the user can use the option code to retrieve a set of data file addresses of records which satisfy complex "and" and "or" conditions.

After one or more calls to GETIND have been executed to retrieve the data file addresses of all records satisfying specified criteria, GETREC can be called to retrieve each record directly via its data file address.

Each call to GETREC causes the next data file address in sequence to be retrieved from the work file; following this, the record itself is retrieved from the data file. There must be one call to GETREC for each record to be retrieved.

For detailed descriptions of the calling arguments to GETIND and GETREC, refer to the descriptions of those subroutines in Section IV.

Descriptor Cards Required

The chart below shows the types of descriptor cards and parameters which must be present on the descriptor card file in a retrieval program. Only those parameters listed in the chart are meaningful for a retrieval job.

Card Type	Parameters
INDEX	FC = isp-index-file-code
INDX2	FC = invff-index-file-code
DATA	FC = data-file-code-1
WORKFL	FC = work-file-code

For a more detailed description of these descriptor cards and descriptor card parameters, refer to Section III.

File Retrieval Subroutines

Below is a summary of the subroutine calls needed in a program to retrieve records from an inverted file:

1. One call to IOPEN to open the file and prepare for the retrieval of records. The syntax of the call to IOPEN is as follows:

```
CALL IOPEN USING record-area, data-file-code-1,
     desc-file-code [,error-code]
```

2. If records are to be retrieved sequentially:

   a. One call to GETSEQ for each record to be retrieved. The syntax of the call to GETSEQ is as follows:

```
CALL GETSEQ USING data-file-code-1, error-code
     [,alt-record-area [,alt-record-size]]
```

   b. Optionally, one call to XREWND whenever it is desired to position the current record pointer to a record having a specific ISP key value. The syntax of the call to XREWND is as follows:

```
CALL XREWND USING data-file-code-1
     [,error-code, function-code]
```

If records are to be retrieved by specification of ISP key value, one call to GETRAN for each record to be retrieved. The syntax of the call to GETRAN is as follows:

```
CALL GETRAN USING data-file-code-1, error-code
     [,alt-record-area ],alt-record-size]]
```

If records are to be retrieved by specification of conditions on inverted key values:

   a. One or more calls to GETIND to retrieve the data file addresses of the records satisfying the specified criteria. The syntax of the call to GETIND is as follows:

```
CALL GETIND USING data-file-code-1,  error-code,
     no-of-recs, option-code, key-no-1,
     rel-code-1, value-1 [,key-no-2,
     rel-code-2, value-2]...

     or

CALL GETIND USING data-file-code-1, error-code,
     no-of-recs, option-code, criteria-table.
```

   b. One call to GETREC for each record to be retrieved. The syntax of the call to GETREC is as follows:

```
CALL GETREC USING data-file-code-1, error-code
    [,alt-record-area ],alt-record-size]]
```

3. One call to ICLOSE to terminate processing of the file. The syntax of the call to ICLOSE is as follows:

```
CALL ICLOSE USING data-file-code-1
```

For a more detailed description of the usage of these subroutines, refer to Section IV.

Temporary Files Needed

When retrieving records from an inverted file via GETIND and GETREC, the is required to provide one temporary file: the work file.

WORK FILE

The work file is the temporary file to which data file addresses are written by GETIND and from which they are retrieved by GETREC.

The work file must be a random file. The formulas given below can be used to calculate the maximum number of links required for the work file. The independent variables used in these formulas are:

NC = total number of conditionals used in all calls to GETIND in the program

NSRn = number of records in the file that satisfy conditional n, where n = 1,2,...,NC

Additionally, let function INT x = largest integer less than or equal to x.

1. Let NSECn denote the number of sectors (64-word blocks) required to hold the data file addresses of all records that satisfy conditional n, where n = 1,2,...,NC. Then

$$NSECn = INT \left[\frac{NSRn + 62}{63} \right]$$

2. If NWS denotes the number of sectors required for the work file, then

$$NWS = 1 + NSEC1 + NSEC2 + NSEC3 + \ldots + NSEC(NC)$$

3. If NWL denotes the number of 3840-word links required for the work file, then

$$NWL = INT \left[\frac{NWS + 59}{60} \right]$$

Memory Allocation and Page Buffering

In a program in which records are retrieved via calls to GETIND and GETREC, the user has very little control over the order in which pages are accessed; the inverted processor does not sort the data file addresses on the work file by page and line number, and the user cannot choose from the work file the data file address from which he would like to retrieve the next record. It is, therefore, very important that an adequate number of page buffers be provided.

If the utilization report from a retrieval program indicates that an unusually high number of physical reads has taken place, the user should increase the amount of memory available for buffer space by increasing the amount of core allocated for .XBUF1 on the $ USE card, and/or by increasing the amount of core allocated for the activity on the $ LIMITS card.

UPDATING AN EXISTING FILE

The term "updating" refers to the deletion, addition, and modification of records. A file is updated by one call to INVUPD, one call to IOPEN, one call to the appropriate routine (ADDREC, CHGREC, or DELREC) for each record to be updated, and one call to ICLOSE.

When a record is added, changed, or deleted, each index (set of coarse and fine index pages) corresponding to a key that is defined for a record of that type, must be updated accordingly.

Adding a Record

When an attempt is made to add a record, via ADDREC, the inverted processor ensures that:

1. No record containing the same ISP key value currently exists on the file.
2. The value contained in the record type field is valid.
3. The record contains no duplicate values for keys defined as UNIQUE.

If any of the above conditions is violated, the record is not stored.

When a record is added, a new inverted index entry is added for each key that is defined for the record. Each new entry must be physically inserted; if there is no room on the page for the entry, then the page must be split; that is, a new page is allocated and half of the entries from the old page are placed on the new page. This shows the importance of specifying at initialization time a reasonable percentage fill for the inverted index pages.

Deleting a Record

When an attempt is made to delete a record, via DELREC, the inverted processor ensures that:

1. A record containing the specified ISP key value does in fact exist on the file.

2. The value contained in the record type field is valid.

If either of the above conditions is violated, no attempt is made to delete the record.

Note that the test for record type validity is not performed at initialization time. Thus, although a record containing an invalid record type field may exist as an inverted file, it is not possible to delete such a record in an inverted file update procedure. The record, however, may be deleted by accessing the file as a simple ISP file (use ISP descriptor cards rather than INVFF) and then calling DELREC.

When a record is deleted, each inverted index entry corresponding to a key that is defined for that record is physically deleted.

Changing a Record

When an attempt to change an existing record, via CHGREC, the inverted processor ensures that:

1. The ISP key value of the record to be replaced does in fact exist on the file.

2. The value contained in the record type field of the record to be replaced is valid.

3. The value contained in the record type field of the record that is to replace the existing record is valid.

4. The record that is to replace the existing record contains no duplicate values for keys defined as UNIQUE.

If any of the above conditions is violated, no attempt is made to replace the record.

When an existing record is to be replaced by a record containing the same ISP key value, the inverted file processor determines which, if any, of the inverted key values in the record are to be changed. For those key values which are to be changed the index entries corresponding to the old values are deleted and index entries corresponding to the new values are inserted.

Buffer Flushing

Buffer flushing is a technique used to insure that file integrity is maintained in the event of a system interruption. It is particularly useful in a transaction processing environment. For a more detailed description of the buffer flushing capability, refer to the current ISP manual.

Descriptor Cards Required

The chart below shows the types of descriptor cards and parameters which must be present on the descriptor card file in an update program. Only those parameters listed in the chart are meaningful for an update job.

Card Type	Parameter
INDEX	FC = isp-index-file-code
INDX2	FC = invff-index-file code
DATA	FC = data-file-code-1

For a more detailed description of these descriptor cards and descriptor card parameters, refer to Section III.

File Update Subroutines

Below is a summary of the subroutine calls needed in a program to update an inverted file.

1. One call or SYMREF to INVUPD - the subroutine itself accomplishes nothing, but the call or SYMREF must be present anywhere in the user program in order to cause the General Loader to load the inverted file update modules. No calling arguments are necessary.

2. One call to IOPEN to open the file and prepare for the addition, deletion, and modification of records. The syntax of the call to IOPEN is as follows:

```
CALL IOPEN USING record-area, data-file-code-1,
     desc-file-code [,error-code]
```

3. One call to ADDREC for each record to be added. The syntax of the call to ADDREC is:

```
CALL ADDREC USING data-file-code-1, error-code
     [,alt-record-size]
```

4. One call to CHGREC for each record to be changed. The syntax of the call to CHGREC is:

```
CALL CHGREC USING data-file-code-1, error-code
```

5. One call to DELREC for each record to be deleted. The syntax of the call to DELREC is;

   CALL DELREC USING data-file-code-1, error-code.

6. One call to XFLUSH for each time the buffers are to be flushed. The syntax of the call to XFLUSH is:

   CALL XFLUSH [USING function-code].

7. One call to ICLOSE to terminate processing of the file. The syntax of the call to ICLOSE is:

   CALL ICLOSE USING data-file-code-1.

For a more detailed description of the usage of these subroutines, refer to Section IV.

Memory Allocation and Page Buffering

The amount of memory available for page buffering can be increased by increasing the amount of core allocated for .XBUF1 on the $ USE card, and/or by increasing the amount of core allocated for the activity on the $ LIMITS card. For a more detailed description of these techniques, refer to the current ISP manual.

FMS Protection and Recovery

The ABORT/ROLLBACK/ option of FMS can be used to prevent incomplete file updating that can result from job or system failure. Details on the FMS file protection feature may be found in the current ISP manual.

Journalization and Restoration

The ISP capability of journalizing changes to a file has not yet been implemented for use by the Inverted File Facility.

UNLOADING AND RELOADING AN INVERTED FILE

The utility program XUTIL may be used to unload and reload an inverted file. This capability allows a user to:

1. Add, delete, or modify inverted key definitions.

2. Remove logically deleted records.

3. Redistribute data records uniformly across the data file; and/or redistribute inverted index entries uniformly across the inverted index file.

4. Correct a descriptor card error.

Unloading an Inverted File

The job setup for a GCOS activity that unloads an inverted file is shown below:

```
$    PROGRAM   XUTIL

$    LIMITS    limit values

$    DATA      .X
ISP  INDEX     FC = isp-index-file-code
ISP  DATA      FC = data-file-code-1
[ISP DATA      FC = data-file-code-2]
 .
 .
 .
$    file      fc,random,ISP index file

$    file      fc,random,primary data file
[$   file      fc,random,additional data file]
 .
 .
 .
$    file      OT,sequential, unloaded file
```

Note that ISP descriptor cards are to be used, rather than INVFF cards. Note also that the inverted index file is not referenced during an unload activity.

Reloading an Inverted File

The job setup for a GCOS activity that reloads an inverted file is shown below:

```
$    PROGRAM   XUTIL

$    USE       .XVOP.,.XVLD.

$    LIMITS    limit values

$    DATA      .X
   .
   . INVFF descriptor cards for a build job
   .
$    file      IN,sequential, file to be loaded or reloaded

$    file      fc,random, ISP index file

$    file      fc,random, inverted index file

$    file      fc,random, primary data file
[$   file      fc,random, additional data file]
 .
 .
 .
```

The $ USE card accomplishes the same purpose as the call or SYMREF to INVBLD in a user-written program, i.e., it allows the inverted file build routines to be loaded.

All rules regarding ISP and inverted key values which apply to a user-written build program also apply to the XUTIL reloading program.

SECTION III

DESCRIPTOR CARD REFERENCE

This section describes, in reference form, the individual types of descriptor card. The descriptions of the card types are arranged alphabetically to facilitate quick reference.

GENERAL FORMAT OF DESCRIPTOR CARDS

One general format applies to all descriptor cards. It is defined as follows:

1	8	16
INVFF	type	parameters

RULES

1. Card type must be one of the folowing words: DATA, ETC, INDEX, INDX2, KEY, RECORD, SORTFL, or WORKFL.

2. Each card must contain at least one parameter. The valid parameters for the different card types are listed with the individual card type descriptions.

3. No blanks may appear within a parameter.

4. Parameters can be separated by a comma and/or one or more blanks.

5. Parameters can appear on a descriptor card in any order.

6. The last parameter on a card can be followed by a blank or comma.

Each type of descriptor card is described below. In the parameter formats, upper case words and equal signs are required. Lower case words represent generic values defined in the description as arguments are to be replaced by specific values. Parameter formats enclosed in brackets are optional and can be omitted.

DATA Descriptor Card

One DATA descriptor card is used to define the primary data file of an inverted file. Additional DATA descriptor cards are used to specify the file codes when there is more than one data file in the inverted file.

PARAMETERS

```
 FC=data-file-code-n      n=1,2,...,8
[PAGESZ=data-page-size]
[PAGEFIL=pct-fill]
[COLCOM]
[BLKSZ=block-size]
```

RULES

1. The argument data-file-code-1 is the GCOS file code of the primary inverted data file. Data-file-code-2 through data-file-code-8 represent the GCOS file codes of up to seven additional data files. The value of data-file-code-n+1, interpreted as a twelve-bit unassigned binary integer, must be one greater than data-file-code-n. The DATA descriptor card for the primary data file may contain any of the parameters. The DATA descriptor card for any additional data file must contain only the FC parameter.

2. The PAGESZ, PAGEFIL, and COLCOM parameters are meaningful only when the DATA descriptor card is used in conjunction with a call to NOPEN or KINIT or when it is used with the XUTIL utility.

3. The argument data-page-size is the number of words in a data file page. This number must be a multiple of 64 between 320 and 4032, inclusive. If the PAGESZ parameter is absent, a value of 320 words is assumed for data-page-size.

4. The argument pct-fill is the percentage of each data file page to be used for the storage of records at the time the file is initially built. If the PAGEFIL parameter is absent, a value of 100 is assumed for pct-fill.

5. The parameter COLCOM signifies that the key values are ordered according to the Commercial Collating Sequence. The absence of this parameter specifies that key values are in the standard collating sequence.

6. The BLKSZ parameter is meaningful only when XUTIL is used to load, unload, or reload an inverted file. The block-size used for reloading a file must be the same size as that used when the file was unloaded. If the BLKSZ parameter is absent, a value of 320 words is assumed.

DATA Descriptor Card

EXAMPLES

```
1         8         16
INVFF     DATA      FC=DF, COLCOM

INVFF     DATA      PAGESZ=448 PAGEFIL=80 FC=F2

INVFF     DATA      FC=AB,PAGESZ=640
INVFF     DATA      FC=AC
INVFF     DATA      FC=AD

INVFF     DATA      FC=DD,PAGESZ=512,BLKSZ=1024
```

More than one data file.
AB is the primary data file.

This card is used as a continuation of a descriptor card of any other type.

PARAMETERS

Any parameters that are valid for the type of descriptor card to be continued.

RULE

This card allows the parameters for a descriptor card of any other type to be placed on separate cards. The rules for the type of card being continued also apply to any ETC cards that follow it.

EXAMPLES

```
1         8         16
INVFF     DATA      FC=XY
INVFF     ETC       PAGESZ=1280
INVFF     ETC       PAGEFIL=75

INVFF     INDEX     PAGESZ=512,
INVFF     ETC       FC=IN
```

INDEX Descriptor Card

This card is used to specify the GCOS file code and the page size of the ISP index file for the inverted file.

PARAMETERS

```
 FC=isp-index-file-code
[PAGESZ=isp-index-page-size]
```

RULES

1. The argument isp-index-file-code is the GCOS file code of the ISP index file for the inverted file.

2. The PAGESZ parameter is meaningful only when this descriptor card is used in conjunction with a call to NOPEN or KINIT or when it is used with the XUTIL utility.

3. The argument isp-index-page-size is the number of words in an ISP index page. This number must ba a multiple of 64 between 320 and 4032, inclusive. If the PAGESZ parameter is absent, a value of 320 words is assumed for isp-index-page-size.

EXAMPLES

```
1         8         16
INVFF     INDEX     FC=IF

INVFF     INDEX     FC=F1, PAGESZ=640

INVFF     INDEX     PAGESZ=384 FC=AA
```

INDX2 Descriptor Card

This card is used to specify the GCOS file code, page size, and percentage fill for the inverted index file for an inverted file.

PARAMETERS

```
 FC=invff-index-file-code
[PAGESZ=invff-index-page-size]
[PAGEFIL=invff-index-pct-fill]
```

RULES

1. The argument invff-index-file-code is the GCOS file code of the inverted index file for the inverted file.

2. The PAGESZ and PAGEFIL parameters are meaningful only when this descriptor card is used in conjunction with a call to NOPEN or KINIT or when it is used with the XUTIL utility.

3. The argument invff-index-page-size is the number of words on an inverted index page. This number must be a multiple of 64 between 320 and 4032, inclusive. If the PAGESZ parameter is absent, a value of 320 words is assumed for invff-index-page-size.

4. The argument invff-index-pct-fill is the percentage of each index file page to be used for the storage of index entries at the time the file is initially built. If the PAGEFIL parameter is absent, a value of 90 percent is assumed for invff-index-pct-fill.

EXAMPLES

```
1         8         16
INVFF     INDX2     FC=IF

INVFF     INDX2     FC=F1, PAGESZ=640

INVFF     INDX2     PAGEFIL=80 PAGESZ=384 FC=AA
```

This card is used to specify the characteristics of a data field to be used as an inverted key field.

PARAMETERS

[RTYPE=record-type] (may be required, see Rule 2)

KEYNO=key-number

KEYSZ=key-size

[KEYOFF=key-offset]

$$\left[\text{NULL} = \left\{\begin{array}{l}\text{null-character}\\ \text{BLANK}\end{array}\right\}\right]$$

[UNIQUE]

[SIGNED]

RULES

1. The KEY card is meaningful only when used in conjunction with NOPEN or KINIT or when it is used with the XUTIL utility.

2. The RTYPE parameter is required and meaningful if and only if a record type field is defined (via the RTYPSZ and RTYPOFF parameters) on the RECORD descriptor card. If a record type field is defined, the argument record-type specifies the value contained in the record type field in all records for which this key field is defined. If no record type field is defined, then all keys are considered to be defined for every record on the file.

3. The argument key-number must be an integer between 0 and 4,095, inclusive. This key-number serves to distinguish the various keys from one another. Key fields from different record types which have the same key-number are defined as participating in the same inverted key.

4. The argument key-size is the size of the key field in characters.

5. The argument key-offset is the number of character positions to the beginning of the key field, measured from the beginning of the record. If the KEYOFF parameter is absent, a value of zero is assumed for key-offset, signifying that the key field is located at the beginning of the record.

6. The NULL parameter signifies that "null processing" is to take place on this key field. The value in the key field will be considered null whenever it consists entirely of the character defined as null-character (when the BLANK option is present, the

space (octal 20) will be considered the null-character). No pointer to null key field values will be written to the inverted index file.

7. The UNIQUE parameter signifies that each new key value to be stored on the data file must be unique, i.e., must not duplicate any key value already present on the file.

8. The SIGNED parameter signifies that the key field in each record contains signed data. The first bit of the field is to be considered the sign bit.

9. There must be one KEY card for each field in each record that is to be considered an inverted key field. One or more keys may be defined for each record type. Moreover, fields from different record types may be defined as the same key.

10. A data field may participate in more than one key; if this is the case, then separate KEY cards must be provided for that field for each key in which it participates.

11. The NULL and UNIQUE parameters are meaningful only insofar as they are present or absent on the first KEY card pertaining to a given key (as determined by key-number).

12. If a record type field is defined, the SIGNED parameter is meaningful only insofar as it is present or absent on the first KEY card pertaining to a given field in a record of a given type.

EXAMPLES

```
1         8         16
INVFF     KEY       RTYPE=1, KEYNO=1, KEYSZ=5, KEYOFF=1,
INVFF     ETC       NULL=BLANK, SIGNED

INVFF     KEY       RTYPE=2 KEYNO=2 KEYSZ=3 KEYOFF=74
INVFF     ETC       NULL=A UNIQUE

INVFF     KEY       RTYPE=3,KEYNO=3,KEYSZ=4,
INVFF     ETC       KEYOFF=7                        } One record type,
INVFF     KEY       RTYPE=3,KEYNO=4,KEYSZ=20,       } multiple keys
INVFF     ETC       KEYOFF=11

INVFF     KEY       RTYPE=4,KEYNO=5,KEYSZ=6,
INVFF     ETC       KEYOFF=31,NULL=BLANK            } Field from different
INVFF     KEY       RTYPE=5,KEYNO=5,KEYSZ=6,        } record types defined
INVFF     ETC       KEYOFF=7                        } as same key
```

KEY Descriptor Card

KEY Descriptor Card

```
1         8         16
INVFF     KEY       RTYPE=6,KEYNO=6,KEYSZ=6,
INVFF     ETC       KEYOFF=11,SIGNED
INVFF     KEY       RTYPE=6,KEYNO=7,KEYSZ=6,
INVFF     ETC       KEYOFF=11
```

Same data field participating in more than one key

This card is used to specify the characteristics of the largest record in an inverted data file.

PARAMETERS

RECSZ=record-size

KEYSZ=key-size

[KEYOFF=key-offset]

[RTYPSZ=record-type-size]

[RTYPOFF=record-type-offset]

RULES

1. These parameters are meaningful only when this descriptor card is used in conjunction with a call to NOPEN or KINIT or when it is used with the XUTIL utility.

2. The argument record-size is the number of words in the largest record in the data file. This value cannot exceed 1024.

3. The argument key-size is the number of characters in the ISP key.

4. The argument key-offset is the number of character positions to the beginning of the ISP key, measured from the beginning of the record. If the KEYOFF parameter is absent, a value of zero is assumed for key-offset, signifying that the ISP key is located at the beginning of the record.

5. The sum of key-offset and key-size cannot exceed 1530.

6. The argument record-type-size is the number of characters in the record type field. This value cannot exceed twelve. If the RTYPSZ parameter is absent, then no record type field is defined for the file.

7. The RTYPOFF parameter is meaningful only if the RTYPSZ parameter is present. The argument record-type-offset is the number of character positions to the beginning of the record type field, measured from the beginning of the record. If the RTYPSZ parameter is present and the RTYPOFF parameter is absent, a value of zero is assumed for record-type-offset, signifying that the record type field is located at the beginning of the record.

8. The sum of record-type-offset and record-type-size cannot exceed 1530.

RECORD Descriptor Card

EXAMPLES

```
1         8         16
INVFF     RECORD    RECSZ=100, KEYSZ=8, RTYPSZ=2

INVFF     RECORD    RTYPSZ=2 RTYPOFF=8 KEYSZ=21 RECSZ=12

INVFF     RECORD    RECSZ=14,KEYSZ=6,KEYOFF=6
```

SORTFL Descriptor Card

This card is used to specify the file code of the temporary file on which entries for the inverted index file are to be written during a build job.

PARAMETER

FC=sort-file-code

RULES

1. The argument sort-file-code must be the file code of a magnetic tape or linked disk file.

2. This descriptor card is meaningful only when used in conjunction with a call to NOPEN or when it is used with the XUTIL utility.

EXAMPLE

```
1         8         16
INVFF     SORTFL    FC=FS
```

WORKFL Descriptor Card

This card is used to specify the file code of the temporary file on which record addresses from the inverted index file are to be written during a retrieval job.

PARAMETER

FC=work-file-code

RULES

1. The argument work-file-code must be the file code of a random disk file.

2. This descriptor card is meaningful only when used in conjunction with a call to IOPEN or LOPEN and is necessary only when records are to be processed via calls to GETIND and GETREC.

EXAMPLE

```
1         8         16
INVFF     WORKFL    FC=WF
```

SECTION IV

SUBROUTINE REFERENCE

This section describes, in reference form, the subroutines needed to process inverted files. The descriptions of the subroutines are arranged alphabetically to facilitate quick reference. Included within the description of each subroutine are the functions performed by each subroutine, the required forms of calling the subroutine, and a list of rules governing the use of the subroutine.

As in the case of ISP, all subroutines are invoked through the standard CALL interface. Therefore, any of the standard programming languages (COBOL, FORTRAN, and GMAP) can be used to process inverted files. Unless otherwise indicated, the CALL formats described in this section are in the form of COBOL statements. Reference should be made to the appropriate programming language manual for specific rules regarding the CALL interface in that language.

CALL FORMAT CONVENTIONS

The CALL formats in this section obey the following conventions:

1. Upper case words are required.

2. Lower case words denote generic arguments that are replaced by specific names in the user's program.

3. Terms enclosed in brackets are optional and can be omitted.

TYPES OF ARGUMENTS

There are three basic types of arguments: location, alphanumeric, and numeric. These types are defined below.

Argument Type	Definition
Location (L)	The location of a block of words in memory.
Alphanumeric (A)	A string of two 6-bit BCD characters, left-justified in a word.
Numeric (N)	A full-word binary integer, defined as INTEGER in FORTRAN and COMPUTATIONAL-1 in COBOL.

All of the arguments that appear in the CALL formats in this section are listed below in alphabetical order.

Argument	Type	Description
alt-record-area	L	The location of an area of memory into which a record is to be placed by GETSEQ, GETRAN, or GETREC. This location may differ from that specified as record-area in the call to IOPEN or LOPEN.
alt-record-size	N	The number of words in the record to be stored by FILINT or ADDREC. Also the number of records to be moved into alt-record-area by GETSEQ, GETRAN, or GETREC. Under certain conditions, ISP stores the size of the record which it has moved into alt-record-area into this location. See the descriptions of GETSEQ, GETRAN, and GETREC for details.
criteria-table	L	The location of an area of memory where GETIND is to find the table of conditionals to be used as criteria for retrieving record addresses.
data-file-code-1	A	The GCOS file code of the primary inverted data file. Under certain conditions, ISP stores the file code in this location. See the descriptions of NOPEN, IOPEN, and LOPEN for details.
desc-file-code	A	The GCOS file code of the file that contains the descriptor cards.
error-code	L	The word into which ISP is to store an error code following a call to ADDREC, CHGREC, DELREC, FILINT, GETIND, GETRAN, GETREC, GETSEQ, IOPEN, LOPEN, NOPEN, XCKPTP, or XREWND. Error codes are described in detail in Section V.
function-code	N	The value that specifies the function to be performed by XREWND.
isp-index-file-code	A	The GCOS file code of the ISP index file.
invff-index-file code	A	The GCOS file code of the inverted index file.
key-no-n	N	Key number of an inverted key being used in a conditional for GETIND. See the description of GETIND in this section for details.
no-of-recs	N	Number of records found by GETIND containing inverted keys which meet specified criteria.

Argument	Type	Description
option-code	N	Value indicating to GETIND the manner in which record addresses retrieved by the current call to GETIND are to be manipulated with the contents of the work file.
record-area	L	The location of an area of memory into which records are read and from which records are written.
rel-code-n	N	Value in a conditional for GETIND indicating how the inverted key field in the data record is to be compared to the specified value value-n. See the description of GETIND in this section for details.
seq-check	L	The location of a block of memory large enough to hold one ISP key value with the same word orientation that it has in a record. The presence of this argument in the first call to FILINT directs ISP to check ISP key values for proper sequence.
sort-file-code	A	The GCOS file code of the file being used as a sort file during a build procedure.
stat-area	L	The location of a seven-word block of memory into which XSTAT is to store the current values of seven file usage parameters. See the description of XSTAT in the current ISP manual.
value-n	L	The location of an area of memory containing the specific value, in a conditional for GETIND, to which values contained in an inverted key field are to be compared. See the description of GETIND in this section for details.
work-file-code	A	The GCOS file code of the file being used as a work file during a retrieval procedure.

Function

Add a new record to an existing inverted file.

Format

```
CALL ADDREC USING data-file-code-1, error-code
     [,alt-record-size]
```

Rules

1. If no CALL or SYMREF to INVUPD is present in the user's program, the call to ADDREC will cause the program to abort with the message "REQUIRED INVERTED FILE SUBROUTINES NOT LOADED".

2. Data-file-code-1 must refer to a file that has been opened by IOPEN or LOPEN.

3. The record to be added to the file must be located starting at record-area, defined at the time IOPEN or LOPEN is called. The ISP key value, record type field (if defined), and inverted key values must be at the positions within this area specified for those fields at initialization time.

4. If the ISP key value of the record to be added already exists in the file, error-code is set to X07.

5. If a record type field is defined for the file, and if the value contained in the record type field was not present as a record-type argument on a KEY descriptor card at initialization time (i.e., if no inverted keys have been defined for a record of that type), then error-code is set to V01.

6. If the value contained in an inverted key field is a duplicate of a key value (for the same key) that is already present on the file, and if that inverted key was defined as UNIQUE, then error-code is set to V02.

7. If the optional argument alt-record-size is specified, its value must be less than or equal to the value of record-size specified at initialization time; otherwise, the option is ignored. If the value of alt-record-size is greater than record-size or zero, or if argument alt-record-size is omitted, then the value of record-size specified at initialization time is used.

Function

Replace an existing record in an inverted file by a record that has the same ISP key value and the same size.

Format

CALL CHGREC USING data-file-code-1, error-code.

Rules

1. If no CALL or SYMREF to INVUPD is present in the user's program, the call to CHGREC will cause the program to abort with the message: "REQUIRED INVERTED FILE SUBROUTINES NOT LOADED".

2. Data-file-code-1 must refer to a file that has been opened by IOPEN or LOPEN.

3. The record that is to replace the existing record must be located starting at record-area, defined at the time IOPEN or LOPEN is called. The ISP key value, record type field (if defined), and inverted key values must be at the positions within this area specified for those fields at initialization time.

4. If the ISP key value of the record does not exist already in the file, error-code is set to X02.

5. If a record type field is defined for the file, and if the value contained in the record type field of either the existing record or the replacement record does not exist as a record-type argument on a KEY descriptor card at initialization time (i.e., if no inverted keys have been defined for a record of that type), then error-code is set to V01.

6. If the value contained in an inverted key field of the record that is to replace the existing record is a duplicate of a key value (for the same key) that is already present on the file, and if that inverted key was defined as UNIQUE, the error-code is set to V02.

Function

Delete a record from an inverted file.

Format

CALL DELREC USING data-file-code-1, error-code.

Rules

1. If no CALL or SYMREF to INVUPD is present in the user's program, the call to DELREC will cause the program to abort with the message "REQUIRED INVERTED FILE SUBROUTINES NOT LOADED".

2. Data-file-code-1 must refer to a file that has been opened by IOPEN or LOPEN.

3. The ISP key value of the record to be deleted must be placed in the ISP key field (defined at initialization time) of record-area, defined at the time IOPEN or LOPEN is called.

4. If the ISP key value of the record to be deleted does not exist in the file, error-code is set to X02.

5. If a record type field is defined for the file, and if the value contained in the record type field of the record to be deleted was not present as a record-type argument on a KEY descriptor card at initialization time (i.e., if no inverted keys have been defined for a record of that type), error-code is set to V01.

Function

Stores a record into an inverted file that has been opened for the first time (by NOPEN or KOPEN).

Format

```
CALL FILINT USING data-file-code-1 [,alt-record-size [,error-code,seq-check]]
```

Rules

1. Data-file-code-1 must refer to a file that has been opened by NOPEN and KOPEN.

2. The record to be stored in the file must be located starting at record-area, defined at the time NOPEN or KOPEN is called. The ISP key value, record type field (if defined), and inverted key values must be at the positions within this area specified for those fields at initialization time.

3. If the optional argument alt-record-size is specified, the record to be stored must be large enough to contain all of the inverted keys defined for a record of its type. If the record is not large enough, the record is not stored and error-code is set to V02. At the same time, however, the value of alt-record-size must be less than or equal to the value of record-size specified at initialization time; otherwise, the option is ignored. If the value of alt-record-size is greater than record-size, or if argument alt-record-size is omitted, then the value of record-size specified at initialization time is used.

4. If argument error-code and seq-check are specified, FILINT will verify that each ISP key value is greater than the preceding ISP key value, according to the collating sequence selected by the user. If an out-of-sequence ISP key is encountered, error code X03 is stored in location error-code. If these two arguments are omitted, no sequence checking is done, and ISP key values are assumed to be in the correct order. Note that if arguments error-code and seq-check are specified, alt-record-size must also be present, although it can have the same value as record-size.

5. If the value contained in an inverted key field of the record to be stored is a duplicate of a key value (for the same key) that is already present on the file, and if that inverted key was defined as UNIQUE, the record is stored in the data file but no entry for the duplicate occurrence is placed in the inverted index file. A warning message to this effect is printed on the utilization report. When a future attempt is made to retrieve via GETIND and GETREC all of the records containing that key value, only the record containing the first occurrence will be retrieved.

Function

Retrieve the data file addresses of all records containing inverted key values which satisfy specified criteria, and either place those addresses on the work file or logically manipulate those addresses with the addresses currently contained on the work file.

Format 1

```
CALL GETIND USING data-file-code-1, error-code, no-of-recs, option-code,
     key-no-1, rel-code-1, value-1 [,key-no-2,rel-code-2, value-2]...
```

Format 2

```
CALL GETIND USING data-file-code-1, error-code, no-of-recs, option-code,
     criteria-table.
```

Rules

1. Data-file-code-1 must refer to a file that has been opened by IOPEN or LOPEN.

2. No-of-recs refers to a location into which GETIND is to store, as a full-word binary integer (FORTRAN INTEGER, COBOL COMPUTATIONAL-1), the number of records on the file which meet the specified criteria. If no records are found which meet the criteria, no-of-recs is set to zero and error-code is set to X02.

3. GETIND obtains the criteria for selecting record addresses to be retrieved by reading a list of conditionals. A conditional is a true-or-false statement about the contents of an inverted key field of a record. This statement is represented in the program logic as a three-element vector consisting of the elements key-no-n, rel-code-n, and value-n, defined as follows:

   Key-no-n - a key-number, as specified on one or more KEY descriptor cards at initialization time, which serves to identify an inverted key.

   rel-code-n - one of the following binary values corresponding to the indicated relational state:

   1 = "greater than"

   2 - "greater than or equal to"

   3 - "equal to"

   4 - "less than or equal to"

   5 - "less than"

   value-n - the specific value to be used for the comparison

The list composed of all such conditionals may be passed to GETIND in either of the following manners:

a. In Format 1, the list of conditionals is a direct part of the CALL format, i.e.:

```
key-no-1, rel-code-1, value-1 [,key-no-2, rel-code-2,
value-2]...
```

b. In Format 2, the argument criteria-table is the location of a table which contains the list of conditionals in the following format:

```
ARG  key-no-1
ARG  rel-code-1
ARG  value-1
[ARG  key-no-2
ARG  rel-code-2
ARG  value-2]...
```

Each conditional is either "true" or "false" with respect to each record in the file. If all of the conditionals are found to be "true" for a given record, the data file address of that record is placed in the active list of record addresses which, when completed, will subsequently be manipulated with the list of record addresses already present on the work file.

4. The argument option-code must be specified in order to indicate the desired action upon the active list of record addresses (those retrieved by the current call to GETIND) with respect to the list which is currently present on the work file (having been placed there by any previous calls to GETIND).

1 - Initialize - replace the current contents of the work file (if any) with the active list. In symbolic notation:

(work file) $\longleftarrow$ (active list)

2 - Logical "or" - replace the list of record addresses currently contained on the work file with the union of that list and the active list. In symbolic notation:

(work file) $\longleftarrow$ (work file) $\vee$ (active list)

3 - Logical "and" - replace the list of record addresses currently contained on the work file with the intersection of that list and the active list. In symbolic notation:

(work file) $\longleftarrow$ (work file) $\wedge$ (active list)

At the time at which the first call to GETIND is executed by the program, option-code must contain the value of one (initialize); if not, error-code is set to X01. If upon any other call to GETIND the value of option-code is out of range (not Equal to one, two, or three), a value of three (logical "and") is assumed.

Function

Retrieve a record that has a specified ISP key value from an inverted file.

Format

```
CALL GETRAN USING data-file-code-1, error-code
     [,alt-record-area [,alt-record-size]].
```

Rule

GETRAN processes an inverted file as a simple ISP file; no reference to the inverted index file is made. Therefore, the rules governing the use of GETRAN with respect to inverted files are the same as those which apply to the use of GETRAN with ISP files. A list of these rules may be found in the description of GETRAN in the current ISP manual.

Function

Retrieve the next available data file address placed on the work file by GETIND; then retrieve the record having that address on the data file.

Format

```
CALL GETREC USING data-file-code-1, error-code
     [,alt-record-area [,alt-record-size]].
```

Rules

1. Data-file-code-1 must refer to a file that has been opened by IOPEN and LOPEN.

2. A call to GETIND must be executed prior to the first call to GETREC.

3. If there are no more data file addresses on the work file remaining to be retrieved, error-code is set to X04.

4. If the record is to be moved to an area other than record-area, defined when IOPEN or LOPEN was called, then the argument alt-record-area must be present to specify the location of this other area.

5. The argument alt-record-size is used to specify the number of words to be moved into alt-record-area. If the value of alt-record-size exceeds the actual size of the record to be moved, then only the number of the words in the record is moved. If the value of alt-record-size is set equal to zero preceding the call to GETREC, then the actual size of the record moved is stored into location alt-record-size by GETREC. If this technique is used, alt-record-size must be set equal to zero preceding every call to GETREC, because it will have been set to a non-zero value by the previous call. Note that if alt-record-size is specified, alt-record-area must also be present.

Function

Retrieves the next available record from an inverted file without reference to the ISP and inverted index files.

Format

```
CALL GETSEQ USING data-file-code-1, error-code
     [,alt-record-area [,alt-record-size]].
```

Rule

GETSEQ processes an inverted file as a simple ISP file; no reference to the inverted index file is made. Therefore, the rules governing the use of GETSEQ with respect to inverted files are the same as those which apply to the use of GETSEQ with ISP files. A list of these rules may be found in the description of GETSEQ in the current ISP manual.

Function

Close an inverted file.

Format

CALL ICLOSE USING data-file-code-1.

Rule

Data-file-code-1 must refer to a file that has been opened by NOPEN, IOPEN, KOPEN, or LOPEN.

Function

Allows inverted file build routines to be loaded.

Format

FORTRAN CALL INVBLD

COBOL CALL INVBLD.

GMAP SYMREF INVBLD

Rule

The INVBLD routine itself accomplishes nothing, but a CALL or SYMREF to INVBLD is necessary for instructing the General Loader to load the inverted file build routines. If no CALL or SYMREF to INVBLD is present and the program attempts to process an inverted file with a call to NOPEN, KOPEN, or KINIT, the procedure will be aborted with the message: "REQUIRED INVERTED FILE SUBROUTINES NOT LOADED".

Function

Allows inverted file update routines to be loaded.

Format

FORTRAN CALL INVUPD

COBOL CALL INVUPD.

GMAP SYMREF INVUPD

Rule

The INVUPD routine itself accomplishes nothing, but a CALL or SYMREF to INVUPD is necessary for instructing the General Loader to load the inverted file update routines. If no CALL or SYMREF to INVUPD is present and the program attempts to process an inverted file with a call to ADDREC, CHGREC, or DELREC, the procedure will be aborted with the message: "REQUIRED INVERTED FILE SUBROUTINES NOT LOADED".

Function

Open an existing inverted file.

Format

CALL IOPEN USING record-area, data-file-code-1, desc-file-code [,error-code].

Rules

1. The following descriptor cards, containing the indicated parameters, must be present on the descriptor card file having file code desc-file-code:

Card Type	Parameters
INDEX	FC = isp-index-file-code
INDX2	FC = invff-index-file-code
DATA	FC = data-file-code-1
WORKFL	FC = work-file-code

   The WORKFL card need be present, however, only if the file is to be processed with calls to GETIND and GETREC. A complete description of all descriptor cards and descriptor card parameters may be found in Section III of this manual.

2. No value for data-file-code-1 need be stored in the user core location data-file-code-1. IOPEN will move the file code of the primary data file, which it finds on the DATA card in the descriptor card file, into core location data-file-code-1.

3. If the optional argument error-code is included, control is returned to the user if any of the error conditions associated with IOPEN (see Section V) are detected. If error-code is omitted and any of these conditions occur, the job will be aborted.

Function

Initialize an inverted file. Designed primarily for use by World Wide Data Management System (WWDMS). The use of KINIT in user programs is discouraged, as NOPEN will accomplish the same function as the combination of KINIT and KOPEN, with greater flexibility.

Format

CALL KINIT USING desc-file-code.

Rule

The following descriptor cards, containing the indicated parameters, must be present on the descriptor card file having file code desc-file-code.

Card Type	Parameters
INDEX	FC = isp-index-file-code PAGESZ = isp-index-page-size (optional, default =320)
DATA	FC = data-file-code-n, n = 1, 2, ... 8 PAGESZ = data-page-size (optional, default = 320) PAGEFIL = pct-fill (optional, default = 100) COLCOM (optional)
RECORD	RECSZ = record-size KEYSZ = key size
RECORD	KEYOFF = key-offset (optional, default = 0) RTYPSZ = record-type-size (optional) RTYPOFF = record-type-offset (optional, default = 0)
INDX2	FC = invff-index-page-size (optional, default = 320) PAGESZ = invff-index-page-size (optional, default = 320) PAGEFIL = invff-pct-fill (optional, default = 90)
KEY	RTYPE = record-type (required if and only if RTYPSZ is present on RECORD card) KEYNO = key-number KEYSZ = key-size KEYOFF = key-offset (optional, default = 0) NULL = { null-character / BLANK } (optional) UNIQUE (optional) SIGNED (optional)

If a record type field is defined (i.e., if the RTYPSZ parameter is present on the RECORD descriptor card), at least one KEY card must be present for each record type (for each unique value contained in the record type field).

A complete description of all descriptor cards and descriptor card parameters may be found in Section III of this manual.

Function

Opens an initialized inverted file for the first time. Designed primarily for use by World Wide Data Management System (WWDMS). The use of KOPEN in user programs is discouraged, as NOPEN will accomplish the same function as the combination of KINIT and KOPEN, with greater flexibility.

Format

CALL KOPEN USING data-file-code-1, isp-index-file-code, invff-index-file-code, sort-file-code, record-area.

Rule

The first four arguments of the CALL format are described in detail in the descriptions of the DATA, INDEX, INDX2, and SORTFL descriptor cards in Section III.

Function

Open an existing inverted file. Designed primarily for use by World Wide Data Management System (WWDMS). The use of LOPEN in user programs is discouraged, as IOPEN will accomplish the same function as LOPEN, with greater flexibility.

Format 1

```
CALL LOPEN USING record-area, data-file-code-1, desc-file-code [,error-code].
```

Format 2

```
CALL LOPEN USING data-file-code-1, isp-index-file-code, invff-index-file-
     code, work-file-code, record-area.
```

Rules

1. Format 1 has the same effect as CALL IOPEN.

2. The first four arguments for Format 2 are described in detail in the descriptions of the DATA, INDEX, INDEX2, and WORKFL descriptor cards in Section III.

Function

Open a new inverted file for the first time.

Format

```
CALL NOPEN USING record-area, data-file-code-1, desc-file-code
     [,error-code].
```

Rules

1. The following descriptor cards, containing the indicated parameters, must be present on the descriptor card file having file code desc-file-code.

Card Type	Parameters
INDEX	FC = isp-index-file-code PAGESZ = isp-index-page-size (optional, default = 320)
DATA	FC = data-file-code-n, n = 1, 2, ... 8 PAGESZ = data-page-size (optional, default = 320) PAGEFIL = pct-fill (optional, default = 100) COLCOM (optional)
RECORD	RECSZ = record-size KEYSZ = key-size KEYOFF = key-offset (optional, default = 0) RTYPSZ = record-type-size (optional) RTYPOFF = record-type-offset (optional, default = 0)
INDX2	FC = invff-index-file-code PAGESZ = invff-index-page-size (optional, default = 320) PAGEFIL = invff-pct-fill (optional, default = 90)
SORTFL	FC = sort-file-code
KEY	RTYPE = record-type (required if and only if RTYPSZ is present on RECORD card) KEYNO = key-number KEYSZ = key-size KEYOFF = key-offset (optional, default = 0) NULL = { null-character / BLANK } (optional) UNIQUE (optional) SIGNED (optional)

If a record type field is defined (i.e., if the RTYPSZ parameter is present on the RECORD descriptor card), at least one KEY card must be present for each record type (for each unique value contained in the record type field).

A complete description of all descriptor cards and descriptor card parameters may be found in Section III of this manual.

2. No value for data-file-code-1 need be stored in the user core location data-file-code-1. NOPEN will move the file code of the primary data file, which it finds on the DATA descriptor card in the descriptor card file, into core location data-file-code-1.

3. If the optional argument error-code is included, control is returned to the user if any of the error conditions associated with NOPEN (see Section V) are detected. If error-code is omitted and any of these conditions occur, the job will be aborted.

Function

Close all inverted and indexed sequential files currently open.

Format

```
CALL WRAPUP.
```

Rules

1. All inverted and indexed sequential files opened by NOPEN, KOPEN, IOPEN, or LOPEN are closed.

2. If unused allocated memory was used for inverted and/or ISP file tables and page buffers, word 37 (octal) in the slave program prefix is restored to the value that it had before ISP altered it on the first call to an open routine.

Function

Establish a checkpoint for all open files.

Format

CALL XCKPTF.

Rule

The rules governing the use of XCKPTF with respect to inverted files are the same as those which apply to the use of XCKPTF with ISP files. A list of these rules may be found in the description of XCKPTF in the current ISP manual.

Function

Establish a checkpoint for the program and for all open files.

Format

CALL XCKPTP [USING error-code].

Rule

The rules governing the use of XCKPTP with respect to inverted files are the same as those which apply to the use of XCKPTP with ISP files. A list of these rules may be found in the description of XCKPTP in the current ISP manual.

Function

Write any updated pages residing in page buffers to random access storage.

Format

CALL XFLUSH [USING function-code].

Rule

The rules governing the use of XFLUSH with respect to inverted files are the same as those which apply to the use of XFLUSH with ISP files. A list of these rules may be found in the description of XFLUSH in the current ISP manual.

Function

Position the current record pointer to the first record in an inverted file or to a record having an ISP key value greater than or equal to a specified ISP key value.

Format

CALL XREWND USING data-file-code-1 [,error-code, function-code].

Rule

XREWND processes an inverted file as a simple ISP file; no reference to the inverted index file is made. Therefore, the rules governing the use of XREWND with respect to inverted files are the same as those which apply to the use of XREWND with ISP files. A list of these rules may be found in the description of XREWND in the current ISP manual.

Function

Restore (rollback) the files that are open at the time of the previous checkpoints to their state at that time. The effect is to cancel all changes to the files since the previous checkpoint.

Format

```
CALL XROLBF.
```

Rule

A call to either XCKPTF or XCKPTP must be executed prior to performing the rollback.

XROLBP XROLPB

Function

Restore (rollback) the program and all files that are open at the time of the previous checkpoint to their state at that time. The effect is to cancel all changes to the files since the checkpoint and to restart the program from that point.

Format

CALL XROLBP.

Rule

A call to XCKPTP must be executed prior to performing the rollback.

Function

Store the current values of seven ISP file attribute parameters.

Format

CALL XSTAT USING data-file-code-1, stat-area.

Rule

XSTAT processes an inverted file as a simple ISP file; no reference to the inverted index file is made. Therefore, the rules governing the use of XSTAT with respect to inverted files are the same as those which apply to the use of XSTAT with ISP files. A list of these rules may be found in the description of XSTAT in the current ISP manual.

SECTION V

ERROR CODES

The call argument error-code, defined in Section IV, identifies the location of a word into which ISP is to store an error code following a call to certain subroutines. This code is stored into the right half of error-code as three 6-bit BCD characters; the left half of error-code is set to zero. The defined error codes, the subroutines that may encounter the corresponding error conditions, and descriptions of the conditions are listed below.

Error Code	Subroutine	Description
000	All	No error encountered.
001-013 (octal)	XCKPTP	Error in taking program checkpoint. Refer to the description of MME GECHEK in the current General Comprehensive Operating Supervisor manual.
X01	GETIND	An attempt has been made to "and" or "or" an active list to a work file list when in fact no list currently exists on the work file.
X02	CHGREC DELREC GETRAN XREWND	A record having the specified ISP key value does not exist in the file.
X02	GETIND	No records containing inverted key values which satisfy the specified criteria were found in the file.
X03	FILINT	During file initialization, a record was encountered with an ISP key value less than an ISP key value already stored in the file.
X04	GETSEQ XREWND	During sequential processing, the ISP end-of-file record was encountered.
X04	GETREC	No more record addresses remain to be retrieved from the work file.
X07	ADDREC	The ISP key value of a new record that is to be added to the file already exists in the file.
X09	NOPEN IOPEN LOPEN	Either the primary data file, ISP index file, or inverted index file is not present.
X10	NOPEN IOPEN LOPEN	Either the primary data file, ISP index file, or inverted index file is not a random file.

Error Code	Subroutine	Description
X11	IOPEN LOPEN	The ISP index file and primary data file do not match; that is, they were not initialized at the same time.
X12	NOPEN IOPEN LOPEN	Either the ISP index file or the primary data file is already open.
V01	ADDREC	No inverted keys are defined for the record type of the record to be added to the file.
V01	CHGREC	No inverted keys are defined either for the record type of the record that is to replace the existing record or for the record type of the existing record.
V01	DELREC	No inverted keys are defined for the record type of the record that is to be deleted.
V02	FILINT	The alternate record size specified by the user is not large enough to contain all of the inverted keys defined for the record to be stored in their data record orientation.
V02	ADDREC	The record that is to be added contains an inverted key value that is a duplicate of a value that is already present on the file, and the key is defined as UNIQUE.
V02	CHGREC	The record that is to replace the existing record contains an inverted key value that is a duplicate of a value that is already present on the file, and the key is defined as UNIQUE.

SECTION VI

ABORT MESSAGES

When ISP encounters an unrecoverable error condition, it places a descriptive message on the execution report and aborts the activity. Some of these error conditions may occur only during the processing of an inverted file, while other may occur during the processing of either an inverted file or a simple ISP file. Only the abort messages specific to inverted file processing are listed in this section. The other messages may be found in the current ISP manual.

In the listing of the abort messages that follow, the lower case letters fc denote an actual file code. The lower case letters nnnnnn denote a numeric value. Depending on the message, this value may represent a record size, a page size, or a page number. The listing of each message is followed by the action that must be taken to correct the error condition.

Message: REQUIRED INVERTED FILE SUBROUTINES NOT LOADED

Action: If the procedure is a build procedure, a call or SYMREF to INVBLD must be present. If the procedure is an update procedure, a call or SYMREF to INVUPD must be present.

Message: UNABLE TO GROW INVFF INDEX FILE 'fc'

Action: Not enough file space has been allocated for the inverted index file. If the inverted file is being built, the size of the inverted index file must be increased and the build job must be run again. If an existing permanent file is being updated, the file must first be unloaded, then the size of the inverted index file must be increased, and finally, the file must be reloaded. Utility XUTIL can be used to unload and reload the file. These procedures are necessary because ISP does not "grow" a permanent file beyond its current size.

Message: NO VALID INVFF RECORDS INPUTTED TO SORT

Action: No records for which inverted keys are defined have been stored by FILINT. Possible causes are:

1) The record type field in each data record does not contain the expected value, or

2) An error has been made in defining the record type field on the RECORD descriptor card, or

3) An error has been made in specifying the pertinent record types on the KEY descriptor card.

If the error condition was caused by a descriptor card error, it is possible to unload the file using the XUTIL utility, correct the descriptor card error, and reload the file via XUTIL.

Message: TOO MANY CALLING ARGUMENTS

Action: It is not possible to include more than 65 arguments in a CALL GETIND statement. Place all of the conditionals in a table and use Format 2 of the CALL GETIND (see the description of GETIND in Section IV).

Message: UNABLE TO GROW INVFF WORK FILE 'fc'

Action: Not enough space has been allocated for the work file. Increase the size of the work file and rerun the program.

SECTION VII

FILE FORMATS

The information contained in this section is presented as an aid to the analyst in the diagnosis of problems related to inverted file processing.

DATA AND ISP INDEX FILE FORMATS

The data and ISP index file formats for an inverted data base are identical to those for a simple ISP data base, with the exception of the following modification to the utilization record format:

Word	Bits	Contents
20	0-17	Effective page size (product of actual page size and percent fill) if data file. Number of words required to hold the ISP key in its data record orientation if index file.
	18-35	Unused

INVERTED INDEX FILE FORMAT

There are seven types of index pages which may be present in an inverted index file. The seven page types may be placed in four categories as follows:

a. Control pages (page types 00 and 01)

b. Key definition pages (page types 02 and 03)

c. Fine index pages (page types 05 and 06)

d. Coarse index pages (page type 04)

The control pages contain file management information and utilization statistics and provide a means for accessing the key definition pages and the coarse and fine index pages.

The key definition pages contain the descriptions of all inverted keys.

The fine index pages contain all of the inverted key values that are present on the file and the data file addresses of all records containing each value. Each page contains information about one key only, although several pages may contain information about the same key.

The coarse index pages, if any exist on the file, contain indices to fine index pages or to other coarse index pages.

All fine index pages are defined as level 01 pages. If more than one fine index page is needed to contain the values and addresses of a given key, a coarse index page of level 02 is constructed to provide an index to all of the level 01 pages for that key. Similarly, if more than one page at level 02

is needed in order to index all level 01 pages for a given key, a level 03 page is constructed to provide an index to all level 02 pages. Consequently, several level 03 pages may be indexed by a level 04 page, and so on. For any given key, there is only one page at the highest level. The highest level page for each key is referenced by an entry on one of the control pages.

The format of each type of inverted index page is shown below.

Control Pages

FIRST CONTROL PAGE - PAGE TYPE 00

Page 1 in the inverted index file is always the first control page and has the following format:

Words	Bits	Contents
1	0-17	Page number (will always be 1)
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (00)
	6-17	Number of keys defined for the file
	18-35	Size in words of inverted index page
3	0-17	Page number of the first reusable inverted index page
	18-35	Total number of reusable inverted index pages
4	0-17	Number of control pages
	18-35	Number of key definition pages
5		Date of initial load (in BCD)
6		Time of initial load (in clock pulses since previous midnight)
7		Date of most recent update (in BCD)
8		Time of most recent update (in clock pulses since previous midnight)
9	0-17	Number of pages allocated for inverted index file
	18-35	Number of pages used in inverted index file

Words	Bits	Contents
10	0-17	Length in characters of longest key
	18-35	Effective page size (product of actual page size and percent fill)
11	0-17	Unused
	18-35	Total number of llinks allocated for inverted index file
12	0-17	Unused
	18-35	Number of control entries on this page
13-16		Control entry #1
17-20		Control entry #2
.		.
.		.
.		.

OTHER CONTROL PAGES - PAGE TYPE 01

All control pages except the first have the following format:

Words	Bits	Contents
1	0-17	Page number
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (01)
	6-17	Page number of this page in sequence with all other control pages
	18-35	Number of control entries on this page
3-6		Control entry #M+1 (where M is the total number of control entries on all previous control pages)
7-10		Control entry #M+2
11-14		Control entry #M+3
.		.
.		.
.		.

CONTROL ENTRY

There exists one control entry for each inverted key defined for the file. Each control entry contains the following information.

Words	Bits	Contents
n	0-17	Page number of the highest level page for this key
	18-35	Highest level number
n+1	0-17	Key number (as defined on KEY descriptor cards)
	18-35	Number of pages in inverted index file containing information about this key (page types 04, 05, 06)
n+2		Number of unique key values in data file for this key
n+3		Total number of key values in data file for this key

Key Definition Pages

FIRST KEY DEFINITION PAGE - PAGE TYPE 02

The first key definition page has the following format:

Words	Bits	Contents
1	0-17	Page number
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (02)
	6-17	Unused
	18-35	Number of words in key definition table
3		EIS descriptor of record type field (=0 if no record type field is defined)
4		EIS descriptor of ISP key field
5	0-17	Zero
	18-35	Number of RECORD entries in key definition table
6	0-17	Word offset to first KEY entry in key definition table, from beginning of table

Words	Bits	Contents
	18-35	Number of KEY entries in key definition table

The remaining words on the page (not counting those words not filled due to the percent fill restriction) are filled with entries from the key definition table. If more key definition entries exist than can fit on one page, the remaining entries are placed on one or more type 03 pages.

OTHER KEY DEFINITION PAGES - PAGE TYPE 03

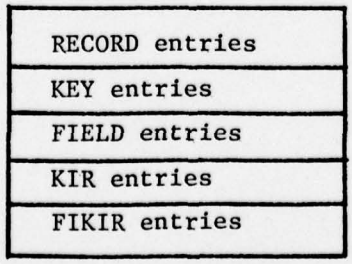
All key definition pages except the first have the following format:

Words	Bits	Contents
1	0-17	Page number
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (03)
	6-17	Page number of this page in sequence with all other key definition pages
	18-35	Unused

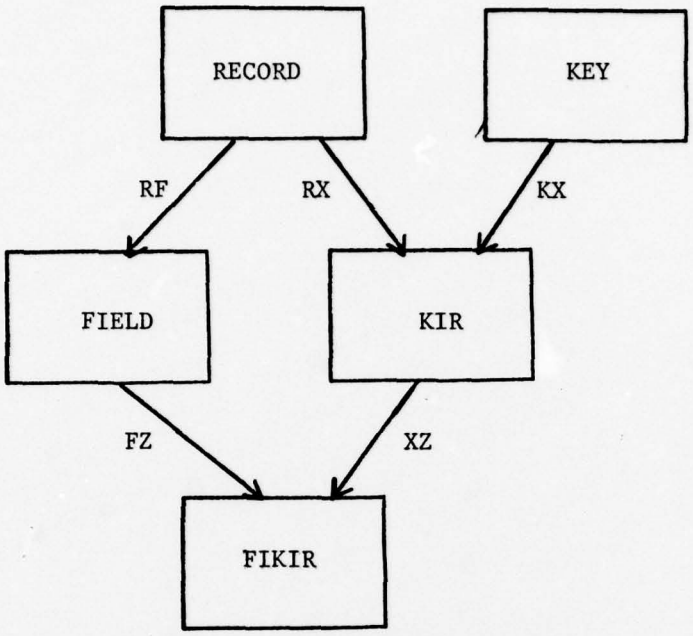
The remaining words on the page (not counting those words not filled due to the percent fill restriction) are filled with entries from the key definition table. If more key definition entries exist than can fit on one page, the table is overflowed onto a subsequent key definition page.

KEY DEFINITION TABLE

The key definition table is a core table which contains the definitions of the inverted key fields and the types of records for which the keys are defined. The table is built when the file is initialized by KINIT or NOPEN. When, in the initialization program, the file is closed by ICLOSE, entries from the key definition table are written to the key definition pages. Each subsequent time the file is opened, the table is read back into core. The key definition table is made up of five types of entries: RECORD, KEY, FIELD, Key In Record (KIR), and Field In Key In Record (FIKIR). The chart below shows the physical positioning of the entries in the table.



The individual entries are logically associated with each other by way of an IDS-like chain structure as shown below:



The chart below shown the notation used in the descriptions of the entry formats:

Notation	Meaning
FZ head	Pointer back to FIELD entry from FIKIR entry*
FZ next	Pointer to next FIKIR entry for specific field*
KX head	Pointer back to KEY entry from KIR entry*
KX next	Pointer to next KIR entry for specific key*
RF head	Pointer vack to RECORD entry from FIELD entry*
RF next	Pointer to next FIELD entry for specific record type*
RX head	Pointer back to RECORD entry from KIR entry*
RX next	Pointer to next KIR entry for specific record type*
XZ head	Pointer back to KIR entry from FIKIR entry*
XZ next	Pointer from KIR entry to associated FIKIR entry*

* Values located in these pointer fields are offsets to the entry from the beginning of the key definition table.

RECORD Entry

There is one RECORD entry in the key definition table for every record type specified on one or more KEY descriptor cards. If there is no record type field defined for the file, then all records are considered to be of the same record type (i.e., record type zero). A RECORD entry is four words in length and contains the following information about a given type of record:

Words	Bits	Contents
1-2		Record type as specified on KEY descriptor cards, left-justified, blank-filled.
3	0-17	Number of words required to hold all inverted keys defined for this type of record in their orientation in the data record.
	18-35	Unused
4	0-17	RF next
	18-35	RX next

KEY Entry

There is one KEY entry in the key definition table for each inverted key defined for the file. A KEY entry is two words in length and contains the following information about a given inverted key:

Words	Bits	Contents
1	0-17	Key number, as specified by the KEYNO parameter on KEY descriptor cards
	18	If bit is on, key is defined as UNIQUE
	19	Unused
	20	If bit is on, a null character is defined for this key
	21-29	Unused
	30-35	If bit 20 is on, these bits contain the null character defined for this key
2	0-17	Key size in characters
	18-35	KX next

FIELD Entry

There is one FIELD entry in the key definition table for each field in each type of record defined as participating in an inverted key. Unless a given field in a given record type participates in more than one key, the number of FIELD entries in the table will be equal to the number of KEY cards on the descriptor file at initialization time. A FIELD entry is three words in length and contains the following information about a given inverted key field:

Words	Bits	Contents
1		EIS descriptor of the field
2	0	If bit is on, field contains SIGNED data
	1-17	Unused
	18-35	FZ next
3	0-17	RF head
	18-35	RF next

Key In Record (KIR) Entry

The Key In Record (KIR) entries serve as connectors between the KEY and the RECORD entries, allowing the inverted processor to find all of the inverted keys associated with a given record type and vice versa. There is one KIR entry in the key definition table for each KEY card on the descriptor file at initialization time. A KIR entry is three words in length and contains the following information:

Words	Bits	Contents
1	0-17	Unused
	18-35	XZ next
2	0-17	RX head
	18-35	RX next
3	0-17	KX head
	18-35	KX next

Field In Key In Record (FIKIR) Entry

The Field In Key In Record (FIKIR) entries serve as connectors between the FIELD and KIR entries, allowing the inverted processor to associate each inverted key field with the keys in which it participates, and vice versa. There is one FIKIR entry in the key definition for each KEY card on the descriptor card at initialization time. A FIKIR entry is two words in length and contains the following pointers:

Words	Bits	Contents
1	0-17	FZ head
	18-35	FZ next
2	0-17	XZ head
	18-35	XZ next

Coarse Index Pages - Page Type 04

Shown below is the format of a coarse index page.

Words	Bits	Contents
1	0-17	Page number
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (04)
	6-17	Key number
	18-35	Unused
3	0-17	Number of entries on this page
	18-35	Number of characters in each entry (size of key field + 3 characters for a page number)
4	0-17	Page number of the page at the next higher level which contains a reference to this page. If this page is the page of the highest level for this key, the field will contain zeroes.
	18-35	Level number of this page

The remaining words on the page (not counting those words not filled due to the percent fill restriction) are filled with type 04 index entries, which are described below. If more entries exist than can fit on one page, then, assuming that the original page has a level number of n ($n \geq 2$), the other entries are placed on subsequent type 04 pages also of level n. A type 04 page of level n+1 is then created to serve as an index to the level n pages for this key.

INDEX ENTRY

Each type 04 index entry for a specific key is k + 3 characters in length, where k is the size in characters of the inverted key field. Assuming the type 04 page on which a certain entry resides has a level number of n ($n \geq 2$), then the field comprised of the last three characters of that entry

contains the page number of a page at level n-1 (note that level 05 and 06 pages have a level number of 1). The first k characters of the entry contain the greatest key value for which an entry may be found on the level n-1 page.

Fine Index Pages

FINE INDEX PAGES FOR UNIQUE KEYS - PAGE TYPE 05

Shown below is the format of a type 05 page.

Words	Bits	Contents
1	0-17	Page number
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (05)
	6-17	Key number
	18-35	Page number of next type 05 page for this key (zero if this is the last page for this key)
3	0-17	Number of entries on this page
	18-35	Number of characters in each entry (size of key field + 6 characters for a data file address)
4	0-17	Page number of the type 04, level 2 page which contains a reference to this page. If there are no type 04 pages for this key, the field will contain zeroes.
	18-35	Unused

The remaining words on the page (not counting those words not filled due to the percent fill restriction) are filled with type 05 index entries, which are described below. If more entries exist than can fit on one page, the other entries are placed one or more subsequent type 05 pages, and a type 04, level 2 page is created to serve as an index to all type 05 pages containing information about the same key.

TYPE 05 INDEX ENTRIES

Each type 05 index entry for a specific key is k+6 characters in length, where k is the size in characters of the inverted key field. The first k characters contain a key value present on the file, while the field comprised of the last 6 characters contains the data file address (line and page number) of the record which contains that key value.

FINE INDEX PAGES FOR NON-UNIQUE KEYS - PAGE TYPE 06

Shown below is the format of a type 06 page.

Words	Bits	Contents
1	0-17	Page number
	18-35	Number of words used on this page, not including the first word
2	0-5	Page type (06)
	6-17	Key number
	18-35	Page number of next type 06 page for this key (zero if this is the last page for this key)
3	0-17	Number of entries on this page
	18-35	Number of characters in each entry (size of key field + 6 characters)
4	0-17	Page number of the type 04, level 2 page which contains a reference to this page. If there are no type 04 pages for this key, the field will contain zeros.
	18	Value continuation bit. If this bit is on, then the data file addresses for the last key value on this page are continued onto another page.
	19-35	Number of data file addresses on this page.

The remaining words on the page are allocated as follows: type 06 index entries are written forward from word 5, and data file addresses are written backward from the end of the page. Any unused words are between the last index entry and the last data file address. The relationship between the index entries and the data file addresses is explained below.

TYPE 06 INDEX ENTRIES

For a key defined as non-unique, each key value may exist in one or more records on the file. Type 06 entries are organized as follows:

1. There exists one index entry corresponding to each distinct key value present on the file (unless bit 18 of word 4 is on, in which case the next type 06 page for the key will contain a "continuation entry" for the key value found in the last entry on the current page).

2. If k is the size of the key in characters, then the first k characters of each entry contain a key value present on the file.

3. The 18-bit field comprised of characters k+4 through k+6 of each entry contains the total number of data file addresses on this page corresponding to the key value in characters 1 through k. The 18-bit field comprised of characters k+1 through k+3 of the entry contains the word offset from word one of the index page to the first (last physically) of such data file addresses.

DATA FILE ADDRESSES

Each data file address occupies one word of storage on the index page and consists of the line number (bits 0-17) and the page number (bits 18-35) of a record on the data file which contains the specified key value.

APPENDIX A

PROGRAMMING EXAMPLES

This appendix consists of an example of a job which utilizes the Inverted File Facility. The job consists of three COBOL programs; the first program builds a new inverted file, the second program retrieves records from the file, and the third program updates the file.

Figure A-1 contains the GCOS control cards and the COBOL source programs for the job. Figure A-2 contains the output produced by the build program. Figure A-3 contains the output produced by the retrieval program. Figure A-4 contains the output produced by the update program.

```
$       IDENT   DMS03,TAMBY                                              000000
$       OPTION  COBOL                                                    000000
$       USE     .XBUF1/4000/,.XBUF2/2/                                   000000
$       COBOL   NDECK                                                    000000
        IDENTIFICATION DIVISION.                                         000000
        PROGRAM-ID. BLDINV.                                              000000
        AUTHOR. S TAMBERRINO.                                            000000
        REMARKS. THIS PROGRAM BUILDS AN INVERTED FILE                    000000
                 FOR USE IN THE INVERTED FILE TEST PACKAGE.              000000
        ENVIRONMENT DIVISION.                                            000001
        CONFIGURATION SECTION.                                           000001
        SOURCE-COMPUTER. 6000 WITH EIS.                                  000001
        OBJECT-COMPUTER. 6000 WITH EIS.                                  000001
        INPUT-OUTPUT SECTION.                                            000001
        FILE-CONTROL.                                                    000001
            SELECT INP-FILE ASSIGN TO IN.                                000001
            SELECT PRT-FILE ASSIGN TO PR FOR LISTING.                    000001
        I-O-CONTROL.                                                     000001
            APPLY STANDARD ON INP-FILE, PRT-FILE.                        000001
        DATA DIVISION.                                                   000002
        FILE SECTION.                                                    000002
        FD  PRT-FILE                                                     000002
            LABEL RECORDS ARE STANDARD                                   000002
            DATA RECORD IS PRT-LINE.                                     000002
        01  PRT-LINE.                                                    000002
            02  PR-1              PICTURE X(10).                        000002
            02  PR-2              PICTURE X(16).                        000002
            02  PR-3              PICTURE X(10).                        000002
            02  PR-4              PICTURE X(84).                        000002
        FD  INP-FILE                                                     000003
            LABEL RECORDS ARE STANDARD                                   000003
            DATA RECORD IS INP-RECORD.                                   000003
        01  INP-RECORD.                                                  000003
            02  EMP-NO            PICTURE 9(4).                         000003
            02  NAME-AND-DATE.                                          000003
                03  NAME-IN       PICTURE X(20).                        000003
                03  DATE-EMP      PICTURE 9(6).                         000003
            02  SALARY            PICTURE 9(4)V99.                      000003
            02  VACATION          PICTURE 9(3).                         000003
            02  SICK              PICTURE S99.                          000004
            02  FILLER            PICTURE X(39).                        000004
        WORKING-STORAGE SECTION.                                         000004
        77  DESC-FILE-CODE        PICTURE XX  VALUE "DC".               000004
        77  DATA-FILE-CODE        PICTURE XX.                           000004
        77  REC-SIZE              PICTURE 9(6) COMPUTATIONAL-1 VALUE 0. 000004
        01  INV-RECORD.                                                  000004
            02  ISP-KEY           PICTURE 9(6) VALUE 0.                 000004
            02  REC-TYPE          PICTURE X.                            000004
            02  EMP-NO-IN-REC     PICTURE 9(4).                         000004
            02  REST-OF-RECORD    PICTURE X(73).                        000005
            02  REST-OF-REC-R REDEFINES REST-OF-RECORD.                 000005
                03  SALARY-R      PICTURE 9(4)V99.                      000005
                03  FILLER        PICTURE X(67).                        000005
        PROCEDURE DIVISION.                                              000005
        010-START.                                                       000005
            OPEN INPUT INP-FILE OUTPUT PRT-FILE.                         000005
```

Figure A-1. Job to Build, Retrieve from, and Update an Inverted File

```
           CALL INVBLD.                                                  000005
           CALL NOPEN USING INV-RECORD, DATA-FILE-CODE, DESC-FILE-CODE.  000005
           MOVE "  ******  " TO PR-1,PR-3.                              000005
           MOVE "RECORD STORED" TO PR-2.                                 000006
       020-PROCESS-CARDS.                                                000006
           READ INP-FILE INTO INP-RECORD                                 000006
               AT END GO TO 030-END-OF-INPUT.                            000006
           MOVE EMP-NO TO EMP-NO-IN-REC.                                 000006
      *                                                                  000006
      *        FORM A TYPE "1" EMPLOYEE NAME RECORD AND STORE IT.        000006
      *                                                                  000006
           ADD 10 TO ISP-KEY.                                            000006
           MOVE "1" TO REC-TYPE.                                         000006
           MOVE NAME-AND-DATE TO REST-OF-RECORD.                         000007
           CALL FILINT USING DATA-FILE-CODE.                             000007
           MOVE INV-RECORD TO PR-4.                                      000007
           WRITE PRT-LINE AFTER ADVANCING 2 LINES.                       000007
      *                                                                  000007
      *        FORM A TYPE "2" VACATION RECORD AND STORE IT.             000007
      *                                                                  000007
           ADD 10 TO ISP-KEY.                                            000007
           MOVE "2" TO REC-TYPE.                                         000007
           MOVE VACATION TO REST-OF-RECORD.                              000007
           CALL FILINT USING DATA-FILE-CODE.                             000008
           MOVE INV-RECORD TO PR-4.                                      000008
           WRITE PRT-LINE AFTER ADVANCING 2 LINES.                       000008
      *                                                                  000008
      *        FORM A TYPE "3" SICK-TIME RECORD AND STORE IT.            000008
      *                                                                  000008
           ADD 10 TO ISP-KEY.                                            000008
           MOVE "3" TO REC-TYPE.                                         000008
           MOVE SICK TO REST-OF-RECORD.                                  000008
           CALL FILINT USING DATA-FILE-CODE.                             000008
           MOVE INV-RECORD TO PR-4.                                      000009
           WRITE PRT-LINE AFTER ADVANCING 2 LINES.                       000009
      *                                                                  000009
      *        FORM A TYPE "4" SALARY RECORD AND STORE IT.               000009
      *                                                                  000009
           ADD 10 TO ISP-KEY.                                            000009
           MOVE "4" TO REC-TYPE.                                         000009
           MOVE SALARY TO SALARY-R.                                      000009
           CALL FILINT USING DATA-FILE-CODE.                             000009
           MOVE INV-RECORD TO PR-4.                                      000009
           WRITE PRT-LINE AFTER ADVANCING 2 LINES.                       000010
           GO TO 020-PROCESS-CARDS.                                      000010
       030-END-OF-INPUT.                                                 000010
           CLOSE INP-FILE PRT-FILE.                                      000010
           CALL ICLOSE USING DATA-FILE-CODE.                             000010
           STOP RUN.                                                     000010
$      EXECUTE DUMP                                                      000010
$      LIMITS  ,30K,-2K                                                  000010
$      DATA    DC                                                        000010
INVFF  INDEX   FC=AA                                                     000010
INVFF  INDX2   FC=CC                                                     000011
INVFF  DATA    FC=BB                                                     000011
INVFF  SORTFL  FC=FS                                                     000011
```

Figure A-1 (Cont.) Job to Build, Retrieve from, and Update an Inverted File

```
INVFF   RECORD  RECSZ=14,KEYSZ=6,RTYPSZ=1,RTYPOFF=6                          000011
INVFF   KEY     KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=1                           000011
INVFF   KEY     KEYNO=2,KEYSZ=20,KEYOFF=11,UNIQUE,RTYPE=1                  000011
INVFF   KEY     KEYNO=3,KEYSZ=6,KEYOFF=31,NULL=BLANK                       000011
INVFF   ETC     RTYPE=1                                                    000011
INVFF   KEY     KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=2                           000011
INVFF   KEY     KEYNO=4,KEYSZ=3,KEYOFF=11,RTYPE=2                          000011
INVFF   KEY     KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=3                           000012
INVFF   KEY     KEYNO=5,KEYSZ=2,KEYOFF=11,SIGNED,RTYPE=3                   000012
INVFF   KEY     KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=4                           000012
$       FILE    AA,X1S,1R                                                  000012
$       FILE    BB,X2S,2R                                                  000012
$       FILE    CC,X3S,2R                                                  000012
$       FILE    S1,X4R,3R                                                  000012
$       FILE    FS,X5R,1L                                                  000012
$       DATA    IN                                                         000012
0200FERGIL SMERD        01057210000020302                                  000012
0201HILRO DEEGO         01157210500012621                                  000013
0202JAH STABO                 11000036558                                  000013
0203AFRE HILL           03067211500041710                                  000013
0204DEMRO TUFF                20000018287                                  000013
0205SAM RILS            03317220500022494                                  000013
0206BACKRUS TUJAL             22000064663                                  000013
0207SMIRNO GILTO        05147223000044389                                  000013
0208FELIX DEMROY              35000075296                                  000013
0209TJLFORD SNUFF       07037237000062882                                  000013
$       SYSOUT  PR                                                         000013
$       BREAK                                                              000014
$       OPTION  COBOL                                                      000014
$       USE     .XBUF1/4000/,.XBUF2/2/                                     000014
$       COBOL   NDECK                                                      000014
        IDENTIFICATION DIVISION.                                           000014
        PROGRAM-ID. INVRET.                                                000014
        REMARKS.   THIS PROGRAM ATTEMPTS TO RETRIEVE ALL OF THE RECORDS    000014
                   FROM THE INVERTED DATA FILE CREATED IN THE PREVIOUS     000014
                   ACTIVITY.                                               000014
        ENVIRONMENT DIVISION.                                              000014
        CONFIGURATION SECTION.                                             000015
        SOURCE-COMPUTER. 6000 WITH EIS.                                    000015
        OBJECT-COMPUTER. 6000 WITH EIS.                                    000015
        INPUT-OUTPUT SECTION.                                              000015
        FILE-CONTROL.                                                      000015
            SELECT PRINT-FILE ASSIGN TO PR FOR LISTING.                    000015
        I-O-CONTROL.                                                       000015
            APPLY STANDARD ON PRINT-FILE.                                  000015
        DATA DIVISION.                                                     000015
        FILE SECTION.                                                      000015
        FD  PRINT-FILE                                                     000016
            LABEL RECORD IS STANDARD                                       000016
            DATA RECORD IS PRINT-LINE.                                     000016
        01  PRINT-LINE.                                                   000016
            02 HEADER-FIELD      PICTURE X(20).                           000016
            02 OUT-RECORD        PICTURE X(84).                           000016
        WORKING-STORAGE SECTION.                                           000016
        77  DESC-FILECODE        PICTURE XX VALUE "DF".                   000016
        77  DATA-FILECODE        PICTURE XX.                              000016
```

Figure A-1 (Cont.) Job to Build, Retrieve from, and Update an Inverted File

```
LSTPU 02  09-16-77   09.919   LSTPU - LISTING OF BCD CARDS NUMBER     01 OF

       77  QUALIFIED-RECORDS   PICTURE 9(6).                                  000016
       77  OPTION-CODE         PICTURE 9(6) VALUE 1.                         000017
       77  KEY-NO-1            PICTURE 9(6) COMPUTATIONAL-1 VALUE 1.         000017
       77  REL-CODE-1          PICTURE 9(6) COMPUTATIONAL-1 VALUE 2.         000017
       77  KEY-VALUE-1         PICTURE X(4) VALUE "0203".                    000017
       77  KEY-NO-2            PICTURE 9(6) COMPUTATIONAL-1 VALUE 3.         000017
       77  REL-CODE-2          PICTURE 9(6) COMPUTATIONAL-1 VALUE 4.         000017
       77  KEY-VALUE-2         PICTURE X(6) VALUE "040172".                  000017
       77  ERROR-CODE          PICTURE X(6).                                 000017
           88 END-O-FILE       VALUE "000X04".                               000017
           88 NO-LISTS         VALUE "000X01".                               000017
       01  INVERT-REC.                                                       000018
           02 ISP-KEY          PICTURE 9(6) VALUE 0.                         000018
           02 REC-TYPE         PICTURE X.                                    000018
           02 EMPLOY-NUMBER    PICTURE 9(4).                                 000018
           02 REST-OF-RECORD   PICTURE X(73).                                000018
       PROCEDURE DIVISION.                                                  000018
       OPEN-FILES.                                                          000018
           OPEN OUTPUT PRINT-FILE.                                          000018
           CALL INVUPD.                                                     000018
           CALL IOPEN USING INVERT-REC,DATA-FILECODE,DESC-FILECODE.         000018
       RETRIEVE-RECORDS.                                                    000019
      *                                                                     000019
      * RETRIEVE THE RECORDS OF ALL EMPLOYEES HAVING AN                     000019
      * EMPLOYEE NUMBER GREATER THAN OR EQUAL TO 0203                       000019
      * AND HIRED BEFORE 040172                                             000019
      *                                                                     000019
           CALL GETIND USING DATA-FILECODE,ERROR-CODE,QUALIFIED-RECORDS,000019
           OPTION-CODE, KEY-NO-1, REL-CODE-1, KEY-VALUE-1,                  000019
           KEY-NO-2, REL-CODE-2, KEY-VALUE-2.                               000019
           IF NO-LISTS GO TO END-UP.                                        000019
       GET-A-RECORD.                                                        000020
           CALL GETREC USING DATA-FILECODE,ERROR-CODE.                      000020
           IF END-O-FILE GO TO END-UP.                                      000020
           MOVE INVERT-REC TO OUT-RECORD.                                   000020
           MOVE "RECORD RETRIEVED" TO HEADER-FIELD.                         000020
           WRITE PRINT-LINE AFTER ADVANCING 2 LINES.                        000020
           GO TO GET-A-RECORD.                                              000020
       END-UP.                                                              000020
           MOVE "END OF FILE" TO HEADER-FIELD.                              000020
           MOVE SPACES TO OUT-RECORD.                                       000020
           WRITE PRINT-LINE AFTER ADVANCING 2 LINES.                        000021
           CLOSE PRINT-FILE.                                                000021
           CALL ICLOSE USING DATA-FILECODE.                                 000021
           STOP RUN.                                                        000021
$      EXECUTE DUMP                                                        000021
$      LIMITS  ,30K,-2K                                                    000021
$      DATA    DF                                                          000021
INVFF  INDEX   FC=AA                                                       000021
INVFF  INDX2   FC=CC                                                       000021
INVFF  DATA    FC=BB                                                       000021
INVFF  WORKFL  FC=WW                                                       000022
$      FILE    AA,X1S,1R                                                   000023
$      FILE    BB,X2S,1R                                                   000023
$      FILE    CC,X3S,2R                                                   000023
$      FILE    WW,X6S,1R                                                   000023
```


Figure A-1 (Cont.) Job to Build, Retrieve from, and Update an Inverted File

```
$       SYSOUT  PR                                                      000023
$       BREAK                                                           000023
$       OPTION  COBOL                                                   000023
$       USE     .XBUF1/4000/,.XBUF2/2/                                  000023
$       COBOL   NDECK                                                   000024
        IDENTIFICATION DIVISION.                                        000024
        PROGRAM-ID. UPDINV.                                             000024
       *                                                                000024
       *  THIS PROGRAM PROCESSES UPDATE TRANSACTIONS AGAINST THE        000024
       *  INVERTED FILE BUILT IN THE PREVIOUS ACTIVITY. A TRANSACTION   000024
       *  CONSISTS OF TWO CARD IMAGES. THE FIRST CONTAINS ONE OF THE    000024
       *  WORDS 'ADD', 'CHANGE', OR 'DELETE', STARTING IN COLUMN 1.     000024
       *  THE SECOND CONTAINS THE ENTIRE NEW RECORD IN THE CASE OF      000024
       *  AN 'ADD' OR 'CHANGE' TRANSACTION, AND A KEY VALUE IN THE      000024
       *  CASE OF A 'DELETE' TRANSACTION. A REPORT IS PRODUCED WHICH    000025
       *  CONTAINS EACH RECORD PROCESSED AND A STATEMENT OF THE ACTION  000025
       *  TAKEN.                                                        000025
       *                                                                000025
        ENVIRONMENT DIVISION.                                           000025
        CONFIGURATION SECTION.                                          000025
        SOURCE-COMPUTER. 6000 WITH EIS.                                 000025
        OBJECT-COMPUTER. 6000 WITH EIS.                                 000025
        INPUT-OUTPUT SECTION.                                           000025
        FILE-CONTROL.                                                   000025
            SELECT INPUT-FILE ASSIGN TO IF.                             000026
            SELECT PRINT-FILE ASSIGN TO PR FOR LISTING.                 000026
        I-O-CONTROL.                                                    000026
            APPLY STANDARD ON INPUT-FILE, PRINT-FILE.                   000026
        DATA DIVISION.                                                  000026
        FILE SECTION.                                                   000026
        FD  INPUT-FILE                                                  000026
            LABEL RECORDS ARE STANDARD                                  000026
            DATA RECORDS ARE INPUT-TRAN, INPUT-DATA.                    000026
        01  INPUT-TRAN.                                                 000026
            02  TRAN-TYPE          PICTURE X(6).                       000027
            02  FILLER             PICTURE X(78).                      000027
        01  INPUT-DATA             PICTURE X(84).                      000027
        FD  PRINT-FILE                                                  000027
            LABEL RECORDS ARE STANDARD                                  000027
            DATA RECORD IS PRINT-LINE.                                  000027
        01  PRINT-LINE.                                                 000027
            02  PRT-1              PICTURE X(10).                      000027
            02  PRT-2              PICTURE X(22).                      000027
            02  PRT-3              PICTURE X(10).                      000027
            02  PRT-4              PICTURE X(84).                      000028
        WORKING-STORAGE SECTION.                                        000028
        77  DESC-FC                PICTURE XX VALUE "DC".              000028
        77  DATA-FC                PICTURE XX.                         000028
        77  TRAN-KEY-WORD          PICTURE X(6).                       000028
            88  ADD-REC            VALUE "ADD".                        000028
            88  CHG-REC            VALUE "CHANGE".                     000028
            88  DEL-REC            VALUE "DELETE".                     000028
        77  ERROR-CODE             PICTURE X(6).                       000028
            88  OKAY               VALUE "000000".                     000028
            88  ISP-KEY-MISSING    VALUE "000X02".                     000029
            88  DUPLICATE-ISP-KEY  VALUE "000X07".                     000029
```

Figure A-1 (Cont.) Job to Build, Retrieve from, and Update an Inverted File

```
           88  INVALID-REC-TYPE VALUE "000V01".                          000029
           88  NON-UNIQUE-INV-KEY  VALUE "000V02".                       000029
       01  INV-RECORD.                                                    000029
           02  ISP-KEY-VALUE   PICTURE 9(6).                             000029
           02  REC-TYPE        PICTURE X.                                000029
           02  REST-OF-RECORD  PICTURE X(77).                            000029
       PROCEDURE DIVISION.                                             000029
       010-START.                                                      000029
           OPEN INPUT INPUT-FILE OUTPUT PRINT-FILE.                    000030
           CALL INVUPD.                                                000030
           CALL IOPEN USING INV-RECORD, DATA-FC, DESC-FC.              000030
           MOVE "  ******  " TO PRT-1, PRT-3.                          000030
       020-READ-TRANS.                                                 000030
           READ INPUT-FILE AT END GO TO 070-END-OF-RUN.                000030
           MOVE TRAN-TYPE TO TRAN-KEY-WORD.                            000030
           READ INPUT-FILE AT END GO TO 070-END-OF-RUN.                000030
           MOVE INPUT-DATA TO INV-RECORD.                              000030
           IF ADD-REC GO TO 030-ADD-A-RECORD.                          000030
           IF CHG-REC GO TO 040-CHANGE-A-RECORD.                       000031
           IF DEL-REC GO TO 050-DELETE-A-RECORD.                       000031
           MOVE TRAN-KEY-WORD TO PRT-4.                                000031
           MOVE "INVALID TRANSACTION" TO PRT-2.                        000031
           GO TO 060-PRINT-A-LINE.                                     000031
       030-ADD-A-RECORD.                                               000031
           CALL ADDREC USING DATA-FC, ERROR-CODE.                      000031
           MOVE "RECORD ADDED" TO PRT-2.                               000031
           MOVE INV-RECORD TO PRT-4.                                   000031
           IF DUPLICATE-ISP-KEY MOVE "DUPLICATE ISP KEY" TO PRT-2.     000031
           IF INVALID-REC-TYPE MOVE "INVALID RECORD TYPE" TO PRT-2.    000032
           IF NON-UNIQUE-INV-KEY MOVE "NON-UNIQUE INV KEY" TO PRT-2.   000032
           GO TO 060-PRINT-A-LINE.                                     000032
       040-CHANGE-A-RECORD.                                            000032
           CALL CHGREC USING DATA-FC, ERROR-CODE.                      000032
           MOVE "RECORD CHANGED" TO PRT-2.                             000032
           MOVE INV-RECORD TO PRT-4.                                   000032
           IF ISP-KEY-MISSING MOVE "ISP KEY NOT FOUND" TO PRT-2.       000032
           IF INVALID-REC-TYPE MOVE "INVALID RECORD TYPE" TO PRT-2.    000032
           IF NON-UNIQUE-INV-KEY MOVE "NON-UNIQUE INV KEY" TO PRT-2.   000032
           GO TO 060-PRINT-A-LINE.                                     000033
       050-DELETE-A-RECORD.                                            000033
           CALL DELREC USING DATA-FC, ERROR-CODE.                      000033
           MOVE "RECORD DELETED" TO PRT-2.                             000033
           MOVE INV-RECORD TO PRT-4.                                   000033
           IF ISP-KEY-MISSING MOVE "ISP KEY NOT FOUND" TO PRT-2.       000033
       060-PRINT-A-LINE.                                               000033
           WRITE PRINT-LINE AFTER ADVANCING 2 LINES.                   000033
           GO TO 020-READ-TRANS.                                       000033
       070-END-OF-RUN.                                                 000033
           CLOSE INPUT-FILE, PRINT-FILE.                               000034
           CALL ICLOSE USING DATA-FC.                                  000034
           STOP RUN.                                                   000034
$      EXECUTE DUMP                                                    000034
$      LIMITS  ,30K,-2K                                                000034
$      SYSOUT  PR                                                      000034
$      FILE    AA,X1S                                                  000034
$      FILE    BB,X2S                                                  000034
```

Figure A-1 (Cont.) Job to Build, Retrieve from, and Update an Inverted File

```
$       FILE    CC,X3S                                          000034
$       DATA    DC                                              000034
INVFF   INDEX   FC=AA                                           000034
INVFF   INDX2   FC=CC                                           000035
INVFF   DATA    FC=BB                                           000035
$       DATA    IF                                              000035
ADD                                                             000035
00049010212TULSA DINGO          090272                          000035
ADD                                                             000035
00041010210JACKFAM BILCH        071372                          000035
CHANGE                                                          000035
00010020202730                                                  000035
DELETE                                                          000035
000210                                                          000036
DELETE                                                          000036
000220                                                          000036
DELETE                                                          000036
000230                                                          000036
DELETE                                                          000036
000240                                                          000036
ADD                                                             000036
00017010204DEMRO TUFF           071377                          000036
ADD                                                             000036
00053X0213MILNER FACTOS         091272                          000037
ADD                                                             000037
00057010214BACKRUS TUJAL                                        000037
CHANGE                                                          000037
00061010251SEYM TEEKO           071377                          000037
CHANGE                                                          000037
00033X0208FELIX DEMROY          071377                          000037
CHANGE                                                          000037
00001010200HILRO DEEGO                                          000037
DELETE                                                          000037
000650                                                          000038
MODIFY                                                          000038
00049010212TULSA DINGO                                          000038
ADD                                                             000038
00045010211ZERO BASTEL                                          000038
ADD                                                             000038
00046020211888                                                  000038
ADD                                                             000038
00047030211999                                                  000038
CHANGE                                                          000038
00013010203STEVE TAMBERRINO                                     000039
$       ENDJOB                                                  000039
```

Figure A-1 (Cont.) Job to Build, Retrieve from, and Update an Inverted File

ISP DESCRIPTOR CARDS ON FILE DC

```
     INVFF  INDEX    FC=AA                                                    00001090
     INVFF  INDX2    FC=CC                                                    00001100
     INVFF  DATA     FC=BB                                                    00001110
     INVFF  SORTFL   FC=FS                                                    00001120
     INVFF  RECORD   RECSZ=14,KEYSZ=6,RTYPSZ=1,RTYPOFF=6                      00001130
     INVFF  KEY      KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=1                         00001140
     INVFF  KEY      KEYNO=2,KEYSZ=2C,KEYOFF=11,UNIQUE,RTYPE=1                00001150
     INVFF  KEY      KEYNO=3,KEYSZ=6,KEYOFF=31,NULL=BLANK                     00001160
     INVFF  ETC      RTYPE=1                                                  00001170
     INVFF  KEY      KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=2                         00001180
     INVFF  KEY      KEYNO=4,KEYSZ=3,KEYOFF=11,RTYPE=2                        00001190
     INVFF  KEY      KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=3                         00001200
     INVFF  KEY      KEYNO=5,KEYSZ=2,KEYOFF=11,SIGNED,RTYPE=3                 00001210
     INVFF  KEY      KEYNO=1,KEYSZ=4,KEYOFF=7,RTYPE=4                         00001220
```

INVERTED FILE PROCESSING REPORT (FILE LOADING PHASE)

KEY NUMBER 1.

NUMBER OF UNIQUE VALUES IS 10.
TOTAL NUMBER OF VALUES IS 40.
NUMBER OF INDEX PAGES REQUIRED IS 1.
NUMBER OF INDEX LEVELS IS 1.

KEY NUMBER 2.

NUMBER OF UNIQUE VALUES IS 10.
TOTAL NUMBER OF VALUES IS 10.
NUMBER OF INDEX PAGES REQUIRED IS 1.
NUMBER OF INDEX LEVELS IS 1.

KEY NUMBER 3.

NUMBER OF UNIQUE VALUES IS 6.
TOTAL NUMBER OF VALUES IS 6.
NUMBER OF INDEX PAGES REQUIRED IS 1.
NUMBER OF INDEX LEVELS IS 1.

KEY NUMBER 4.

NUMBER OF UNIQUE VALUES IS 10.
TOTAL NUMBER OF VALUES IS 10.
NUMBER OF INDEX PAGES REQUIRED IS 1.
NUMBER OF INDEX LEVELS IS 1.

KEY NUMBER 5.

NUMBER OF UNIQUE VALUES IS 10.

Figure A-2. Output from Inverted File Build Program

TOTAL NUMBER OF VALUES IS 10.
NUMBER OF INDEX PAGES REQUIRED IS 1.
NUMBER OF INDEX LEVELS IS 1.

INDEXED SEQUENTIAL PROCESSOR UTILIZATION REPORT

DATA FILE BB		INDEX FILE AA		FILE ATTRIBUTES		JOB ATTRIBUTES	
LOGICAL READS	0	PHYSICAL READS	0	MAXIMUM RECORD SIZE (WORDS)	14	BUFFER SIZE (WORDS)	320
LOGICAL WRITES	40	PHYSICAL WRITES	2	KEY SIZE (CHARACTERS)	6	NUMBER OF BUFFERS	9
PHYSICAL READS	2	PAGE SIZE (WORDS)	320	KEY OFFSET (CHARACTERS)	0	FILE ACCESS	BUILD
PHYSICAL WRITES	4	PAGES ALLOCATED	12	COLLATING SEQUENCE	6000		
PAGE SIZE (WORDS)	320	COARSE PAGES	0	FILE INITIALIZED	09/06/77 09.98		
PAGES ALLOCATED	24	FINE PAGES	1	FILE LAST UPDATED	09/06/77 09.99		
PAGES USED INITIALLY	3	TOTAL PAGES USED	1	DELETED RECORDS	0		
OVERFLOW PAGES USED	0			OVERFLOW RECORDS	0		
TOTAL PAGES USED	3			TOTAL RECORDS	40		

SNJ48 = 1995T, ACTIVITY # = 02, , REPORT CODE = 51, RECORD COUNT = 000041

```
******  RECORD STORED     ******  000010102006FERGIL SMERD        011572
******  RECORD STORED     ******  00002020200203
******  RECORD STORED     ******  00003030200002
******  RECORD STORED     ******  00004040200100000
******  RECORD STORED     ******  00005010201HILRD DEEGO          011572
******  RECORD STORED     ******  00006020201126
******  RECORD STORED     ******  00007030201121
******  RECORD STORED     ******  00008040201105000
******  RECORD STORED     ******  00009010202JAH STABO
******  RECORD STORED     ******  00010020202365
******  RECORD STORED     ******  00011030202S8
******  RECORD STORED     ******  00012040202110000
******  RECORD STORED     ******  00013010203AFRE HILL            030672
******  RECORD STORED     ******  00014020203417
******  RECORD STORED     ******  00015030203I0
******  RECORD STORED     ******  00016040203115000
******  RECORD STORED     ******  00017010204DEMRO TUFF
******  RECORD STORED     ******  00018020204182
```

Figure A-2 (Cont.) Output from Inverted File Build Program

```
****** RECORD STORED ****** 000190302048[illegible]
****** RECORD STORED ****** 000200402042000000
****** RECORD STORED ****** 00021010205SAM RILS          033172
****** RECORD STORED ****** 0002202025224
****** RECORD STORED ****** 0002303020594
****** RECORD STORED ****** 000240402052050000
****** RECORD STORED ****** 00025010206BACKRUS TUJAL
****** RECORD STORED ****** 0002602020646
****** RECORD STORED ****** 0002703020663
****** RECORD STORED ****** 000280402062200000
****** RECORD STORED ****** 00029010207SMIRNO GILTO      051472
****** RECORD STORED ****** 0003002020743
****** RECORD STORED ****** 0003103020789
****** RECORD STORED ****** 000320402072300000
****** RECORD STORED ****** 00033010208FELIX DEMROY
****** RECORD STORED ****** 0003402020875 2
****** RECORD STORED ****** 0003503020896
****** RECORD STORED ****** 000360402083500000
****** RECORD STORED ****** 00037010209TULFORD SNUFF     070372
****** RECORD STORED ****** 0003802020962 8
****** RECORD STORED ****** 0003903020982
****** RECORD STORED ****** 000400402093700000
SNUMB = 1995T, ACTIVITY # = 03, , REPORT CODE = 74, RECORD COUNT = 000140
```


Figure A-2 (Cont.) Output from Inverted File Build Program

```
ISP DESCRIPTOR CARDS ON FILE DF

     INVFF  INDEX    FC=AA                                                  00002170
     INVFF  INDX2    FC=CC                                                  00002180
     INVFF  DATA     FC=BB                                                  00002190
     INVFF  WORKFL   FC=WW                                                  00002210

                            INDEXED SEQUENTIAL PROCESSOR UTILIZATION REPORT

       DATA FILE BB                 INDEX FILE AA                 FILE ATTRIBUTES                        JOB ATTRIBUTES

 LOGICAL READS              2   PHYSICAL READS        1   MAXIMUM RECORD SIZE (WORDS)      14   BUFFER SIZE (WORDS)    320
 LOGICAL WRITES             0   PHYSICAL WRITES       0   KEY SIZE (CHARACTERS)             6   NUMBER OF BUFFERS       11
 PHYSICAL READS             3   PAGE SIZE (WORDS)   320   KEY OFFSET (CHARACTERS)           0   FILE ACCESS      RETRIEVAL
 PHYSICAL WRITES            0   PAGES ALLOCATED      12   COLLATING SEQUENCE             6000
 PAGE SIZE (WORDS)        320   COARSE PAGES          0   FILE INITIALIZED    09/06/77 09.98
 PAGES ALLOCATED           24   FINE PAGES            1   FILE LAST UPDATED   09/06/77 09.99
 PAGES USED INITIALLY       3   TOTAL PAGES USED      1   DELETED RECORDS                   0
 OVERFLOW PAGES USED        0                             OVERFLOW RECORDS                  0
 TOTAL PAGES USED           3                             TOTAL RECORDS                    40


 SNUMB = 1995T, ACTIVITY # = 04, , REPORT CODE = 51, RECORD COUNT = 000004


RECORD RETRIEVED     0001301020 3AFRE HILL          030672

RECORD RETRIEVED     0002101020 5SAM RILS           033172

END OF FILE

 SNUMB = 1995T, ACTIVITY # = 05, , REPORT CODE = 74, RECORD COUNT = 000192
```


Figure A-3. Output from Inverted File Retrieval Program

ISP DESCRIPTOR CARDS ON FILE DC

```
    INVFF  INDEX   FC=AA                          00003490
    INVFF  INDX2   FC=CC                          00003500
    INVFF  DATA    FC=BB                          00003510
```


INDEXED SEQUENTIAL PROCESSOR UTILIZATION REPORT

DATA FILE BB		INDEX FILE AA		FILE ATTRIBUTES		JOB ATTRIBUTES	
LOGICAL READS	0	PHYSICAL READS	1	MAXIMUM RECORD SIZE (WORDS)	14	BUFFER SIZE (WORDS)	320
LOGICAL WRITES	11	PHYSICAL WRITES	1	KEY SIZE (CHARACTERS)	6	NUMBER OF BUFFERS	11
PHYSICAL READS	4	PAGE SIZE (WORDS)	320	KEY OFFSET (CHARACTERS)	0	FILE ACCESS	UPDATE
PHYSICAL WRITES	4	PAGES ALLOCATED	12	COLLATING SEQUENCE	6000		
PAGE SIZE (WORDS)	320	COARSE PAGES	0	FILE INITIALIZED	09/06/77 09.98		
PAGES ALLOCATED	24	FINE PAGES	1	FILE LAST UPDATED	09/06/77 10.00		
PAGES USED INITIALLY	3	TOTAL PAGES USED	1	DELETED RECORDS	4		
OVERFLOW PAGES USED	0			OVERFLOW RECORDS	0		
TOTAL PAGES USED	3			TOTAL RECORDS	45		

SNJ43 = 1995T, ACTIVITY # = 06, , REPORT CODE = 51, RECORD COUNT = 000020

```
******  RECORD ADDED            ******  000490102l2TULSA DINGO        090272          00003540
******  RECORD ADDED            ******  000410102l0JACKFAM BILCH      071372          00003560
******  RECORD CHANGED          ******  000100202730                                  00003580
******  RECORD DELETED          ******  000210                                        00003600
******  RECORD DELETED          ******  000220                                        00003620
******  RECORD DELETED          ******  000230                                        00003640
******  RECORD DELETED          ******  000240                                        00003660
******  DUPLICATE ISP KEY       ******  000170102040EMRO TUFF         071377          00003680
******  INVALID RECORD TYPE     ******  000530X0213MILNER FACTOS      081272          00003700
******  NON-UNIQUE INV KEY      ******  000570102l4BACKRUS TUJA_                      00003720
******  ISP KEY NOT FOUND       ******  000610102l5SEYM TEEKO         071377          00003740
******  INVALID RECORD TYPE     ******  000330X0208FELIX DEMROY       071377          00003760
******  NON-UNIQUE INV KEY      ******  000010102l0FILRO DEEGO                        00003780
******  ISP KEY NOT FOUND       ******  000650                                        00003800
******  INVALID TRANSACTION     ******  MODIFY
******  RECORD ADDED            ******  000450102llZERO BASTEL                        00003840
```

Figure A-4. Output from Inverted File Update Program

AD-A074 081 PLANNING RESEARCH CORP MCLEAN VA F/G 5/2
INVERTED FILE FACILITY USER'S GUIDE.(U)
AUG 77 S TAMBERRINO DCA100-73-C-0015
UNCLASSIFIED PRC-706-22 NL

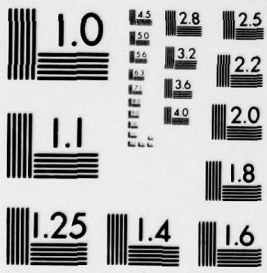
2 OF 2
AD A074081

END
DATE FILMED
10-79
DDC

OF 2

D

074081



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Figure A-

```
RECORD CHANGED      000130102033STEVE TAMBERRINO
RECORD ADDED        000470302121199
RECORD ADDED
```


Figure A-4 (Cont.) Output from Inverted File Update Program